МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра комп'ютерних
систем та мереж

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт з дисципліни

ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

для студентів спеціальності 123 «Комп'ютерна інженерія»
усіх форм навчання
(частина 2)

Тернопіль, 2018

Осухівська Г.М. Методичні вказівки для виконання лабораторних робіт з дисципліни «Технології проектування комп'ютерних систем» для студентів спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Частина 2. / Укладачі Г.М. Осухівська, Н.Р. Шаблій – Тернопіль: ТНТУ ім. І. Пулюя, 2018. – 86 с.

Укладачі: доц., к.т.н., Осухівська Г.М.
асист. Шаблій Н.Р.

Рецензент доцент кафедри КБ, к.т.н., доц. Козак Р.О.

Відповідальний
за випуск: зав. каф КС, к.т.н., доц. Осухівська Г.М.

Методичні вказівки розглянуті і затверджені на засіданні кафедри комп'ютерних систем та мереж протокол № 3 від 14.11.2018 р.

Методичні вказівки розглянуті і затверджені на засіданні науково-методичної ради факультету комп'ютерно-інформаційних систем і програмної інженерії, протокол № ____ від _____ р.

Методичні вказівки складено з урахуванням матеріалів літературних джерел, наведених у переліку.

З М І С Т

Вступ

Дисципліна «Технології проектування комп’ютерних систем» відноситься до однієї з основних дисциплін природничо-наукового циклу у підготовці фахівців спеціальності 123 «Комп’ютерна інженерія». Вона орієнтована на прикладні аспекти і охоплює питання, які пов’язані із застосуванням систем автоматизованого проектування при розробці комп’ютерних систем

Метою викладання дисципліни «Технології проектування комп'ютерних систем» є вивчення принципів автоматизованого проектування комп'ютерних систем, методів та алгоритмів, які використовуються на різних етапах проектування комп’ютерних систем, засвоєння основних прийомів і методів проектування, які використовуються при розробці сучасних комп’ютерних систем.

На базі здобутих під час вивчення дисципліни знань та умінь, фахівцем вирішуватимуться основні професійні задачі, які потребують розуміння принципів автоматизованого проектування і застосування систем автоматизованого проектування для розробки комп’ютерних систем Матеріал курсу використовується перш за все для виконання дипломної роботи.

Лабораторні роботи виконуються для закріплення теоретичних знань, отриманих студентами під час лекційних занять та самопідготовки. Метою виконання лабораторних робіт є вироблення у студентів навичок та вмінь самостійної роботи при проектуванні комп’ютерних систем з використанням автоматизованих систем.

Метою вказівок є допомога студентам при підготовці та виконанні лабораторних робіт за 5 темами, які увійшли до цього видання

У збірнику наведено опис кожної лабораторної роботи. Структурно матеріал до кожної лабораторної роботи включає тему та мету роботи, короткі теоретичні відомості, порядок виконання роботи, вимоги до оформлення звіту, контрольні питання, перелік літературних джерел

Методичні вказівки забезпечують можливість студентам самостійно підготуватись до виконання лабораторних робіт, а викладачеві звести до мінімуму вступні пояснення.

Лабораторна робота № 6

Вивчення мови проектування VHDL

Мета роботи: ознайомитись із мовою проектування VHDL на прикладі програми ModelSim

Теоретичні відомості

Одним із важливих етапів проектування засобів комп'ютерної техніки є розробка електричних схем. Ця робота пов'язана з великими затратами праці, контролем правильності й відповідності задуманому проекту, необхідністю чіткого опису створених схем, труднощами з їхнім супроводом і модернізацією. САПР мають засоби введення і редагування схем.

Мова Very high speed integrated circuits Hardware Description Language (VHDL) була розроблена в 1983 р. на замовлення Міністерства оборони США з метою формального опису логічних схем для всіх етапів розробки електронних систем, починаючи з модулів мікросхем і закінчуючи великими обчислювальними системами. Вона є стандартною мовою з 1987 р. Поряд з мовою Verilog вона є базовою мовою при розробці сучасних комп'ютерних систем.

За допомогою VHDL простіше й швидше ввести й перевірити великий проект, яким часто і є розробка комп'ютерної системи. Десятьма рядками VHDL можна описати як 1, так і 100000 тригерів. Мікросхему з інтеграцією більше 10000 вентилів розробити тільки за допомогою електричних схем дуже важко через громіздкість схем.

Проектування великих обчислювальних пристроїв (ОП).

Проект на VHDL – об'єднання структури системи і алгоритму її функціонування. Для системи, описаної на VHDL, необов'язково виконувати перевірку правильності її функціонування, наприклад, шляхом її макетування. Щоб визначити, чи правильно система чи її частина виконує заданий алгоритм, достатньо VHDL-програму запустити на виконання в симуляторі VHDL. Відповідні САПР перетворять VHDL-опис у комплект документації для виготовлення пристрою.

Проект на VHDL – самодокументований, тобто він не вимагає додаткового технічного опису або опису у вигляді схем. Нечіткість і недбалість опису виключаються, тому що проект на VHDL нескладно перевірити.

Висока надійність проекту. Синтаксичний аналіз, моделювання й компіляція в логічну схему швидко виявляють помилки проекту.

Проект на VHDL – універсальний проект. Розроблений один раз обчислювальний чи будь-який інший блок комп'ютерної системи може бути використаний у багатьох інших проектах. При цьому багато структурних і функціональних параметрів блоку можуть бути настроюваними (параметри розрядності, об'єму пам'яті, елементна база, склад блоку й структура міжелементних з'єднань).

Проект на VHDL – портативний проект. Розроблений для однієї елементної бази, проект легко переноситься на іншу елементну базу, наприклад, НВІС із різною технологією.

Проект на VHDL – проект, що довго живе. Електрична схема завжди розробляється під конкретні елементну базу та інтерфейс. Відповідно до того, що елементна база змінюється за період 2-5 років, за цей же період застарівають і електричні схеми, що її використовують. Проект на VHDL може бути повторно використаний через кілька років. Гарне технічне рішення (наприклад, винахід), описане на VHDL, може бути корисним протягом десятиліть.

VHDL - універсальний засіб опису комп'ютерних систем на рівнях:

- алгоритмічному,
- структурному,
- регістрових передач (RTL) і потоків даних (dataflow),
- логічному,
- аналогових схем.

Хід програмування з використанням VHDL

На рисунку 1 показана схема розробки проекту обчислювального пристрою (ОП) комп'ютерної системи, призначеного для виконання на базі програмованої логічної інтегральної схеми (ПЛІС)

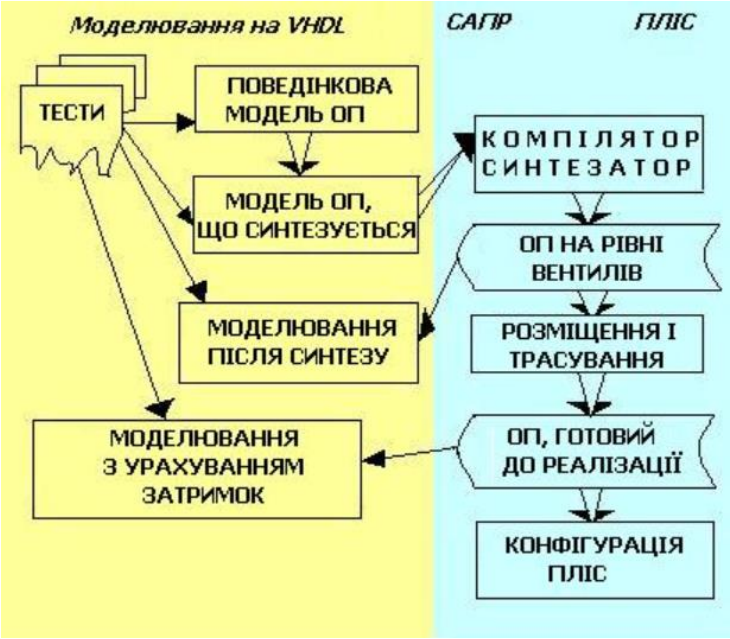


Рис. 1. Схема розробки проекту обчислювального пристрою

Спочатку ОП описується у вигляді своєї поведінкової моделі, на якій відпрацьовується задуманий алгоритм функціонування ОП. Потім ця модель вручну переробляється в синтезовану модель ОП, описану на рівні регістрових передач. Така модель, будучи трансльованою компілятором-синтезатором, дає проектну документацію у вигляді файлу опису схеми ОП на рівні вентилів (EDIF-файл). При цьому автоматично виконується логічна оптимізація ОП. Одночасно цей файл автоматично перетвориться в VHDL-модель ОП на рівні вентилів.

Проект ОП у вигляді Electronic Distribution International Format (EDIF) – файлу приймається як вхідний всіма САПР виготовлення ПЛІС і НВІС. Ці САПР виконують заміну вентилів на бібліотечні компоненти, їхнє розміщення на площі кристала, трасування міжелементних з’єднань, проектування масок, перевірку відповідності проектним нормам і т.п. У результаті записуються файли проектної документації виготовлення кристала і його логічної моделі, що враховує затримки як у вентилях, так і в міжелементних з’єднаннях. Ця модель також представляється на VHDL.

Вартість помилок при проектуванні НВІС дуже висока, особливо на ранніх етапах. Тому всі етапи проектування - алгоритмічний, структурний, логічний, технологічний - супроводжуються моделюванням ОП за допомогою, так званого тестуючого стенду (testbench). Цей стенд представляє VHDL-модель, складовими частинами якої є модель ОП, що тестується, і моделі генератора тестових сигналів і логічного аналізатора,

що перевіряють правильність функціонування ОП. Причому на всіх етапах може використовуватися той самий тестуючий стенд і ті самі тестові файли.

З яких причин VHDL використовується в сучасних САПР мікроелектроніки?

Історично склалося, що в мікроелектронній індустрії найбільше поширення здобула мова Verilog. Півтора десятиліття тому ця мова виграла конкурентну боротьбу з іншими мовами завдання ОП, завдяки невеликим необхідним обчислювальним ресурсам колишніх робочих станцій і досить точним результатам моделювання НВІС. VHDL - більш універсальна й гнучка мова, але вона програвала у швидкодії мові Verilog, особливо при моделюванні на рівні вентилів і транзисторів. VHDL отримала широке поширення в університетах і дослідницьких установах, тому що це строга, стрункa, універсальна й розширювана мова. Так, наприклад, з'явилися пакети VHDL для аналогового моделювання, моделювання багатозначної логіки. Крім того, симулятори VHDL були набагато дешевше симуляторів Verilog.

Всі сучасні САПР мікроелектроніки мають компілятори як з Verilog, так і з VHDL. Програміст, що освоїв VHDL, без особливих зусиль може перейти до програмування мовою Verilog. Але не навпаки.

Найважливішими якостями VHDL у САПР виступають наступні:

Гнучкість. Проект, описаний на VHDL, може бути легко настроєний під конкретні завдання споживача.

Універсальна мова. VHDL – загальноприйнята мова для всіх основних фірм-виготовлювачів мікросхем ПЛІС, ПЛМ, замовних НВІС, як стандартна мова для завдання складних проектів. Проектування з VHDL – стійка тенденція в інженерній технології. Існують компілятори, що транслюють VHDL- програми в еквівалентні їм Verilog – програми.

Моделювання з урахуванням затримок. Фірми-виготовлювачі мікросхем у своїх САПР забезпечують генерацію моделей результатів розміщення і трасування, описаних на VHDL.

Стандартне підключення блоків. Конструкції мови, такі як entity, port map, configuration, забезпечують надійне й швидке стикування блоків, які розроблені різними фірмами й розробниками, у різному сполученні.

Стандартне тестування. На всіх етапах розробки виконується тестування за однією методикою тими самими тестами.

VHDL - стандарт майбутнього. Всі нові САПР засновані на технології трансляції опису ОП мовою опису апаратури. Використання VHDL – гарантія того, що через 5 і 10 років знайдеться САПР, що підтримує старі розробки.

Хід проектування

Нижче на ведено приклад програми, що підраховує кількість одиниць у вхідному образі. Номери рядків не є частиною коду VHDL.

```
1.  LIBRARY ieee;
2.  USE ieee.std_logic_1164.ALL;
3.  ENTITY count IS
4.    GENERIC (w : positive := 8);
5.    PORT (a  : IN  std_logic_vector(w-1 DOWNTO 0);
6.          q : OUT integer RANGE 0 TO w);
7.  END count;
8.
9.  ARCHITECTURE behaviour OF count IS
10.   FUNCTION cnt (a:std_logic_vector) RETURN integer IS
11.     VARIABLE nmb : INTEGER RANGE 0 TO a'LENGTH;
12.   BEGIN
13.     nmb := 0;
14.     FOR i IN a'RANGE LOOP
15.       IF a(i)='1' THEN nmb:=nmb+1; END IF;
16.     END LOOP;
17.     RETURN nmb;
18.   END cnt;
19. BEGIN
20.   q <= cnt(a);
21. END behaviour;
```

Рис. 2. Приклад програми, яка підраховує кількість одиниць у вхідному образі

Змінна *w* у 4 рядку є сталою зі значенням 8. На вхід *a* подається сигнал у двійковому коді. Вихід *q* є цілим числом і знаходиться в діапазоні між 0 та *w*.

Є багато шляхів, щоб підрахувати кількість одиниць в масиві. Функція, яка наведена у прикладі, в якості вхідних об'єктів має *std_logic_vector* – є необмеженим масивом; довжина цього типу (ще) не відома!

6.1 Аналіз / Компіляція

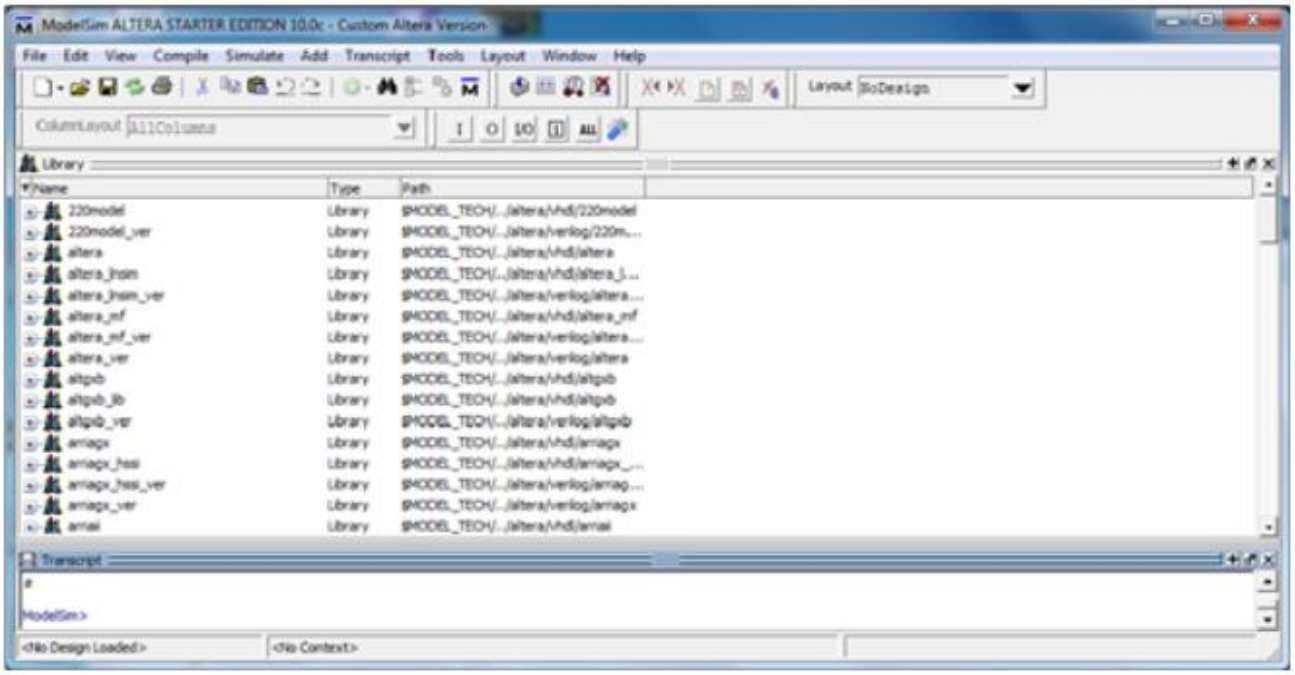


Рис. 3. Вікно програми *ModelSim*

При запуску програми*ModelSim*відкривається вікно яке показано на рис. 2. Для початку створюємо проект: *File→New→Project*

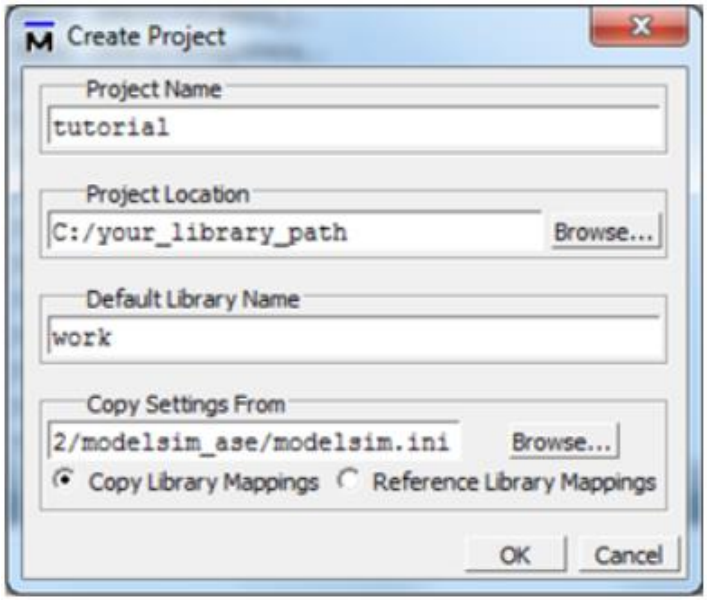


Рис. 4. Вікно запуску програми ModelSim

Виберіть «Ім'я проекту» і «місце проекту». Бібліотеку за замовчуванням не має бути змінено, а також виберіть опцію "копіювати бібліотечні відображення".

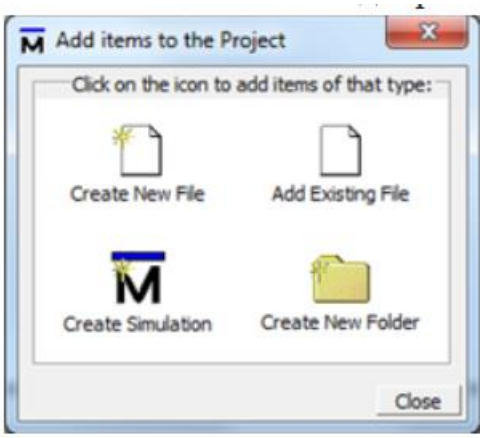


Рис. 5. Вікно бібліотеки ModelSim

Виберіть «додати існуючий файл» і перейдіть в каталог з вихідним файлом і виберіть файл "Counter.vhd".

Натисніть «ОК», та закрийте вікно «add items to the Project»

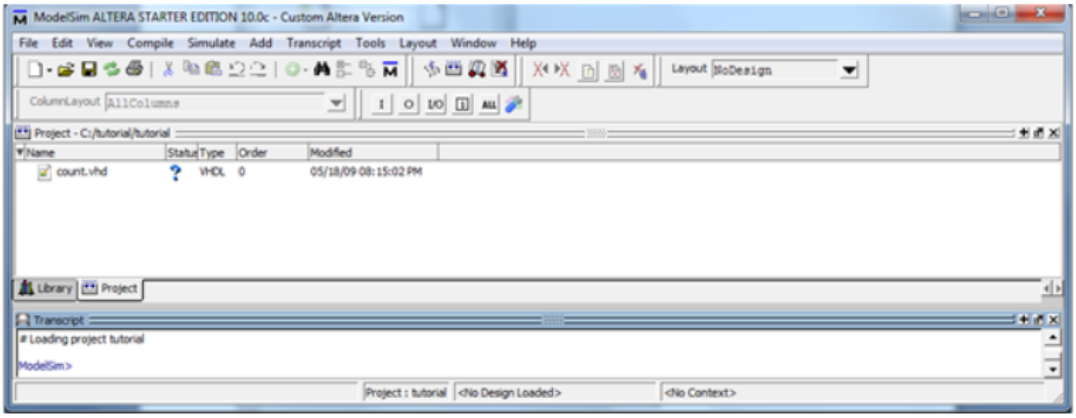


Рис. 6. Основне вікно ModelSim

Виберіть файл *“count.vhd”* → *compile* → *compile selected*.

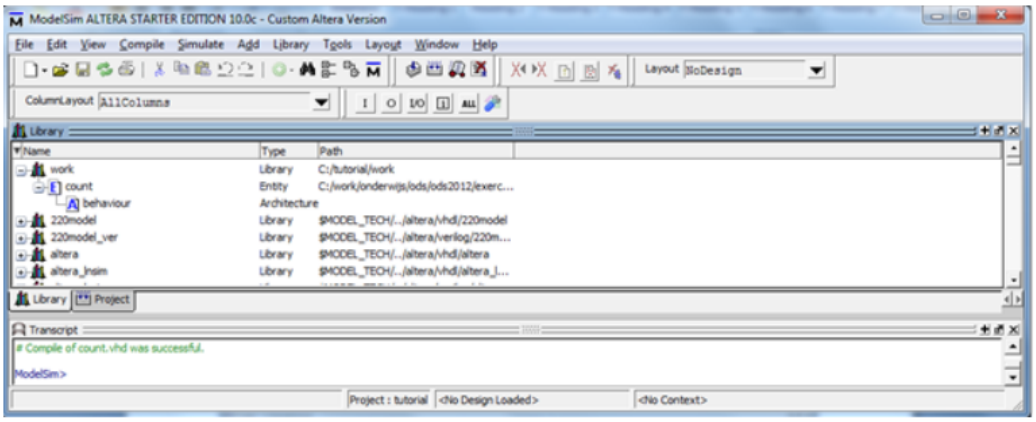


Рис. 7. Вигляд вікна після компіляції

У головному вікні показано бібліотеки, а у нижнє – («Transcript») – використовується для команд, що вводяться користувачем, і для представлення інформації користувачеві. Можна закріпити (як показано вище) і відкріпити (окреме вікно), використовуючи стрілку в правому верхньому куті вікна.

За замовчуванням всі дані VHDL зберігаються в бібліотеці роботи.

Зміст роботи бібліотеки управляється ModelSim. Змінювати вміст цієї бібліотеки за межами ModelSim та зберігати вихідні файли в цій бібліотеці заборонено! Для видалення скомпільованого файлу з бібліотеки необхідно клацнути правою кнопкою миші на проекті (вкладка бібліотеки повинна бути обрана) в графічному інтерфейсі і обрати «Видалити». У такому випадку видаляється вихідний файл та все ще знаходиться в проекті. Для видалення файлу з проекту необхідно клацнути правою кнопкою миші на файлі (вкладка проекту повинна бути обрана) в графічному інтерфейсі і обрати «Видалити з проекту». Важливо: якщо файл був компільований то підрозділи, які були в файлі не видаляються з бібліотеки!

ModelSim використовує термін Compile замість Analysis. Для сигналу VHDL об'єкта ModelSim використовує термін *object*. Для змінної VHDL об'єкта ModelSim використовує термін *local*. Під час налагодження *locals* не відображається за замовчуванням. Щоб внести зміни: вкладка Вид ® виберіть «*locals*».

6.2 Імітація

Натисніть правою клавішою миші на назву архітектури ‘behaviour’, і ви можете завантажити свій дизайн в симуляторі (або можна скористатись меню *simulate* → *start simulation*).

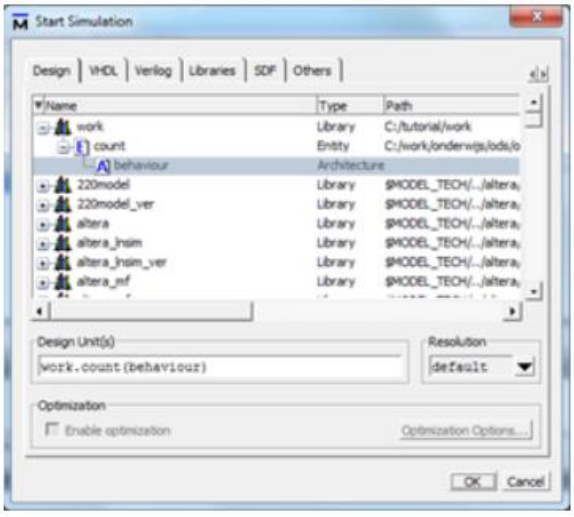


Рис. 8. Вибір проекту, який повинен бути змодельований

У ModelSim-Altera Starter Edition опція "Enable optimization" відключена. Іноді ліцензійна версія ModelSim / QuestaSim використовує оптимізацію лише за замовчуванням. Оптимізація підвищує швидкість моделювання, але під час налагодження не всі сигнали і варіації видимі. **Тому виберіть "full visibility" у вкладці Параметри оптимізації.**

Виберіть оформлення та натисніть «ОК».

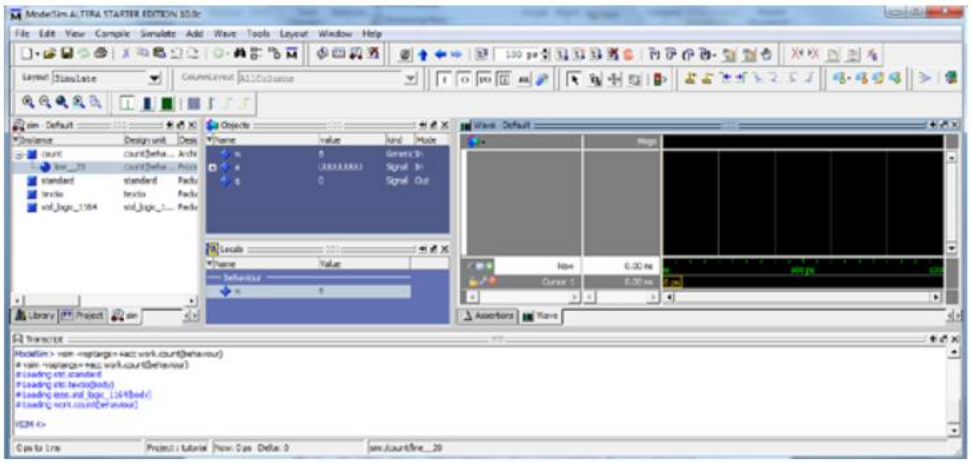


Рис. 9. Вигляд головного вікна проекту ModelSim

Під час моделювання для того щоб побачити деякі сигнали необхідно ввести:

*add wave * <return>*

(замість * ви маєте ввести список з іменами сигналів, розділених комами).

Якщо сигнал *a* та *q* не показані у вікні *vawe*, ймовірно, не вибрано *count* у вікні *instance*. Виберіть *count* і повторіть *add wave **. Вікна *objects* та *locals* показують відповідно сигнали та змінні які видно у *instance*. Оберавши *line__20* також можна побачити генерування *w*. Також *objects* та *locals* можна перетягувати до вікна *wave*.

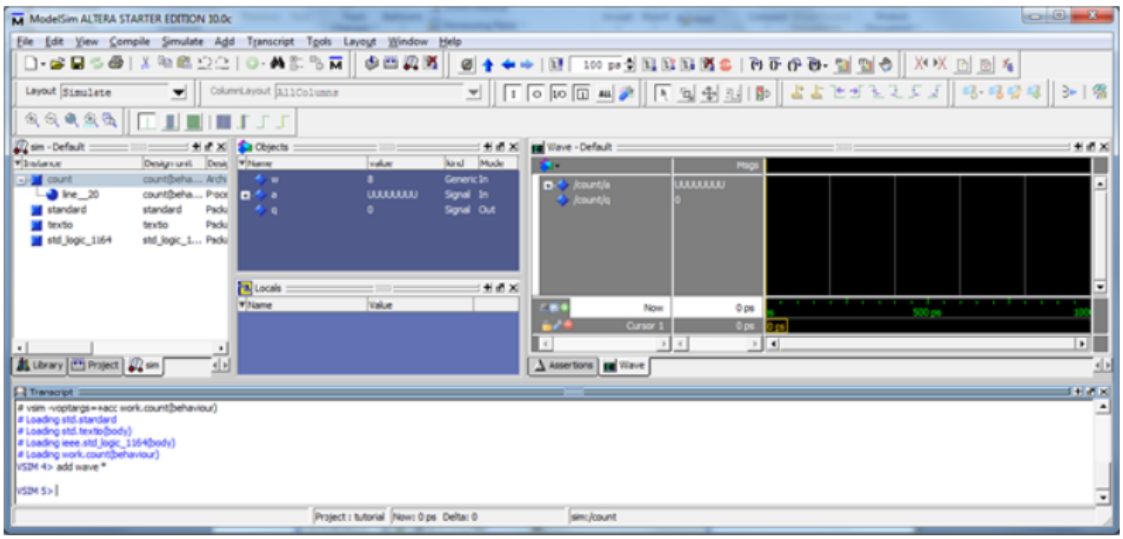


Рис. 10. Вибір проекту, який повинен бути змодельований

За допомогою команди *run* виконується симулювання
run 200ns <enter>
Чому рівне значення ‘U’?
За допомогою команди можна застосовувати вхідний шаблон для *a*:
force a 01100011 <enter>
run 100ns <enter>
Можна призначити кілька значень для вхідних даних з допомогою:
force a 11111111, 00111111 10ns, 11110101 20ns <enter>
run 100ns <enter

У описі VHDL обов’язково потрібно ставити пропуск між цифрою та розмірністю часу. Таким чином, запис *«100ns»* не є коректним. Правильно буде *«100 ns»*. ModelSim видасть помилку:

Warning: [4] <path>.<file>(<line number>): (vcom-1207) An abstract literal and an identifier must have a separator between them.

Така помилка можлива тому, що приклади у перших VHD стандартах (1987) пропуску не було.

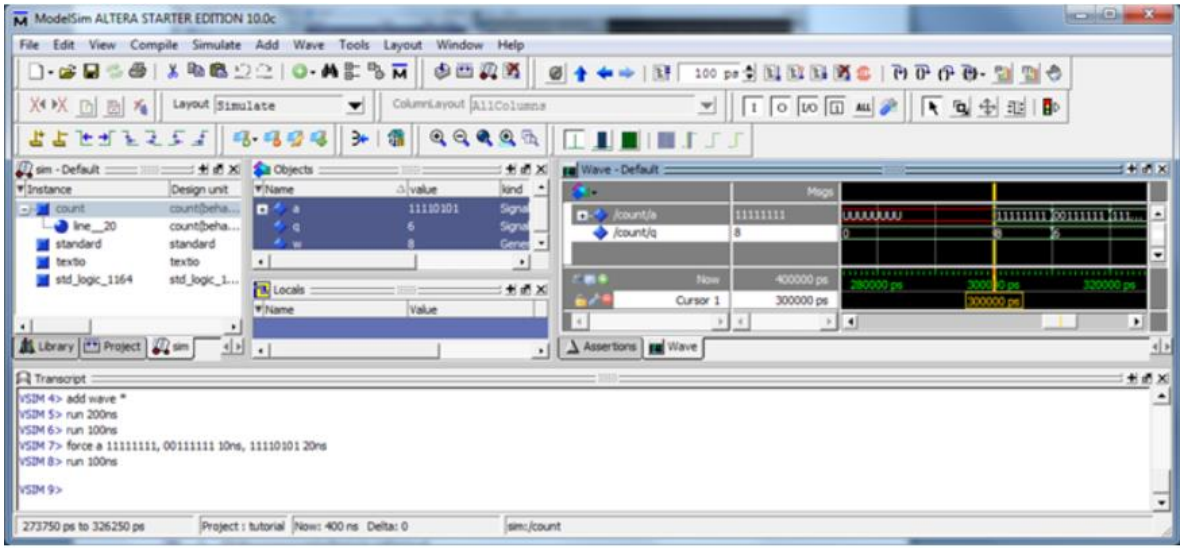


Рис. 11. Вибір проекту, який повинен бути змодельований після внесення коректив

6.2.1 Script-файли зі стимулами

Від вибору інструменту залежить застосовування стимулів та використання Script-файлів. Розглянемо простий приклад. Створимо файл “demo.do” з наступним вмістом:

```
force a 00011111
run 100 ns
force a 10100000
run 100 ns
```

У ModelSim ці Script-файли виконуються з допомогою команди:

do demo.do <return>

1. При синхронному програмуванні необхідний тактовий сигнал. Припустимо, сигнал clk це хронологічна лінія. Повторення шаблону викликається з допомогою команди:

force clk 0, 1 50 ns –repeat 100 ns

(команда clk може використовуватись також для інших сигналів, не лише для годинника)

2. У ModelSim команда ‘run–all’ починає імітацію і зупиняється лише тоді, коли, немає запланованих подій. Ця команда не використовується, коли тактовий сигнал генерується за методом описаним у пункті 1.

3. Команди, які вводяться у вікні стенограми можна записати у файл за допомогою команди “write transcript <filename>”. Цей файл може бути використаний як Script-файл згодом.

6.2.2 Стимуляція генерування з VHDL

Від застосування стимулів, як представлено в попередньому розділі, залежить вибір інструменту. Також можливо, і наполегливо рекомендується, щоб бути інструментально незалежним, генерувати стимули використовуючи VHDL.

Пошук тестових даних для проектування не є легким завданням. У наступному прикладі наведено тест, в якому ширина вектору даних не є надто великою.

Далі наведений простий набір тестів. Він містить один оператор процесу. Спочатку генерує всі нулі, чекає 10 нс, тоді генерує всі одиниці і знову чекає. Циклічний параметр *i* (який неявно оголошується!) прямує від 0 до 2^w-1. Це ціле число перетвориться в бітовий шаблон (з використанням двійкового кодування;). Для перетворення використовується функція to_unsigned. Ця функція перетворює ціле значення *i* без використання вектора з довжиною *w*. Ця функція знаходиться в пакеті numeric_std (цей пакет знаходиться в бібліотеці IEEE). Однак у випадку, якщо загальний (~ константа) *w* велика це завдання буде трудомістким. Тому в даному прикладі цикл закінчився у разі коли *i* дорівнює 20. Процес закінчується очікування. Це означає, що процес не буде відновлювати виконання.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY testset IS
  GENERIC (w : positive := 8);
  PORT (data : OUT std_logic_vector(w-1 DOWNTO 0));
END testset;
ARCHITECTURE set1 OF testset IS
BEGIN
  PROCESS
  BEGIN
      data <= (others => '0'); -- all zero
      WAIT FOR 10 ns;

      data <= (others => '1'); -- all one
      WAIT FOR 10 ns;

      FOR i IN 0 to 2**w-1 LOOP
          data <= std_logic_vector(to_unsigned(i,w));
          EXIT WHEN i=20; -- an exhaustive test is not performed if w is large
          WAIT FOR 10 ns;
      END LOOP;
      WAIT; -- forever
  END PROCESS;
END set1;
```

Довідкова інформація: numeric_std

Пакет numeric_std заявляє два типи:

- Знаковий (двійкове подання) і
- Без знаковий (бінарне подання).

Обидва типи схожий на тип std_logic_vector; масиви з елементами типу std_logic.

```
variable sa,sb,sc : signed(2 downto 0);
variable ua,ub,uc : unsigned(2 downto 0);
variable a,b,c : std_logic_vector(2 downto 0);
```

Якщо *sa* призначено '111' то значення інтерпретується як -1 (двійкове подання)
Якщо *ua* призначено '111' то значення інтерпретується як 7
Якщо *a* призначено '111' то жодні значення інтерпретації не пов'язані з ним!

Що є результатом судження: *sa* := *sb* + "11"
Операнди не мають однакову довжину. Оскільки *sb*

Що стосується операнда без знакового типу, найкоротший вектор доповнюється нулями.
Що стосується операнда типу std_logic_vector, на можливо виконати додавання тому, що жоден номер інтерпретації не пов'язаний з цим типом.
VHDL є строго універсальна мова, тому не можна писати:

```
a := sa;
```

Однак типи тісно пов'язані. У цьому випадку для функції перетворення типу можуть бути використані:

```
a := std_logic_vector(sa);
```

Для цілого значення вектору введіть наступне:

```
integer_value := to_integer(sa);
```

Якщо необхідно перетворити ціле число у векторі треба додати довжину вектора:

```
sa := to_signed(integer_value,3) or
us := to_unsigned(integer_value,us'LENGTH)
```

Атрибут **LENGTH** використовується для наведеного вище прикладу.

6.2.2.1. Додавання блоку розробки у проект

Якщо симулятор як і раніше активний, то завершіть поточне моделювання з допомогою меню імітації.

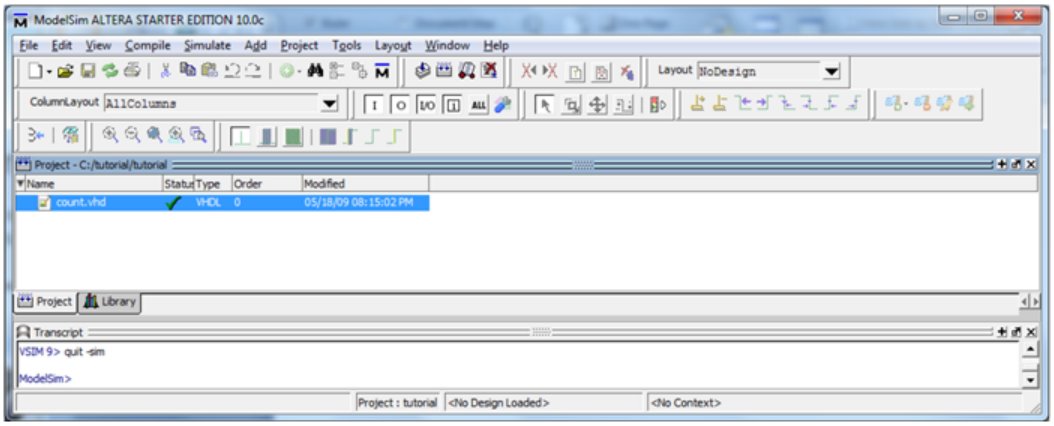


Рис. 12. Вікно завершення моделювання

Щоб додати *testset.vhd* до проекту, виконайте наступні дії:

1. Переконайтесь, що кнопка ***project*** вибрана.
2. Клацніть правою кнопкою миші у вікні проекту.
3. Перейдіть до project→Existing file
4. Натисніть ОК

Компілюйте розробку і виконайте симулювання: run –all <enter>

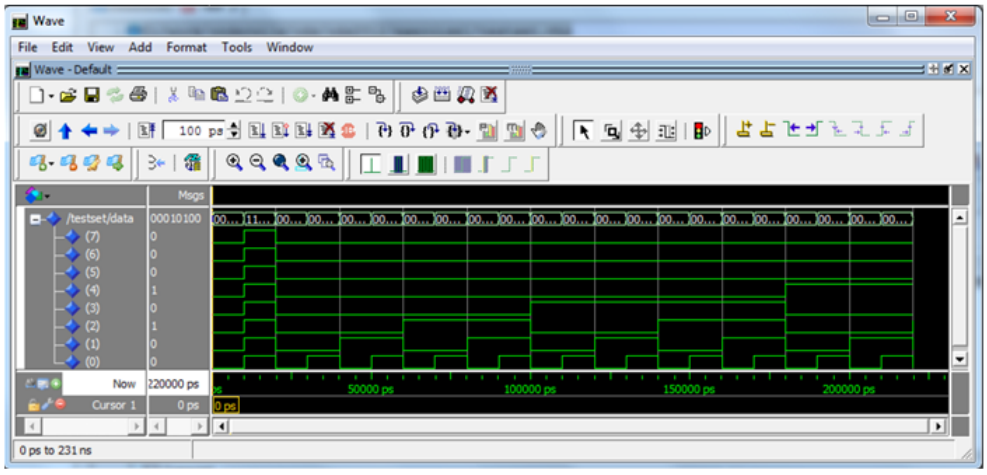


Рис.13. Результат симулювання тестового набору

На рисунку 14 показано структурний опис VHDL, довжина шаблону має крок 10.

Рис.14. Приклад тестової програми

Рис.14. Конструкція ієрархії показана у верхньому лівому вікні

За допомогою цих кнопок можна покроково пройти по проекту (наприклад, щоб знайти помилки).

Часто використовується кнопка «крок» . В такому випадку виконується тільки один оператор (одночасно або послідовно). Також відкрите вікно документа, в якому показано (стрілкою), який наступний оператор буде виконаний.

Під час усунення проблеми часто необхідно зупинити програму в певній точці і виконати налагодження перед цією точкою. Потрібно двічі клацнути на номері рядка який з’являється у місці зупинки курсора.

Рис. 15. Вигляд вікна під час усунення неполадок

6.3. Імітаційна модель

У VHDL можна одночасно опрацьовувати кілька паралельних заяв. Їх порядок не має ніякого впливу на поведінку.

Паралельні заяви можуть «спілкуватися» тільки один з одним, використовуючи сигнали[1]. Якщо привласнити значення сигналу, то значення сигналу не оновлюється миттєво. Це означає, що всі процеси будуть використовувати одні й ті ж значення сигналу; отже, моделювання порядку не залежить від одночасності заяви. Якщо привласнити значення змінної, то змінна оновлюється одразу.

Якщо записати

```
y <= a after 10 ns;
```

то, вихідний сигнал *y* слідує за вхідним сигналом *a* із затримкою 10 нс. (Точніше сигнал *a* має бути стабільним протягом 10 нс)

Якщо записати

```
y <= a
```

то, вихід сигнал оновлюється після затримки дельти. Затримки не відображаються у вікні. Може бути нескінченне число затримок дельта. ModelSim видасть попереджувальне повідомлення, коли виконає 1000 кроків дельта.

ModelSim показує результат сигналів після стадії дельта. Повторіть попереднє моделювання використовуючи наступну команду:

```
add list * <return>.
```

---

[1] Можна використовувати "загальну змінну", але це не є частиною цієї навчальної програми, оскільки результат моделювання залежить від інструментів синтезу.

Завдання на лабораторну роботу № 6

1. Ознайомитись із теоретичними відомостями.
2. Ознайомитись із роботою програми ModelSim
3. Створити проект у програмі ModelSim на мові VHDL.
4. Змоделювати імітаційну модель вузла схеми відповідно до заданого варіанту
5. Результати моделювання у вигляді скріншотів представити у звіті.

Лабораторна робота № 6

Вивчення середовища проектування комп'ютерних систем Quartus II

Мета роботи: ознайомлення із середовищем проектування Quartus II. Створення проекту.

Теоретичні відомості

Програмний пакет Quartus® II фірми Altera® являє собою повне, багатоплатформне середовище проектування, легко адаптується до вимог конкретного проекту. Це комплексне середовище для розробки систем на програмованому кристалі (SOPC). Пакет Quartus II включає в себе всі утиліти, необхідні для роботи з мікросхемами FPGA і CPLD. На рисунку 1 показані основні етапи проектування в середовищі Quartus II.

Наступна послідовність дій описує типовий порядок проектування в середовищі **Quartus II** з допомогою графічного інтерфейсу користувача:

1. Створення нового проекту, вказання типу використовуваної мікросхеми чи сімейства з допомогою команди **New Project Wizard** (меню **File**).
2. Створення вихідного файлу проекту на мовах **Verilog HDL, VHDL**, чи **Altera Hardware Description Language** (**AHDL**) з допомогою текстового редактора (**Text Editor**). Окрім того, можна створити блок-схему проекту в графічному редакторі (**Block Editor**) використовуючи символи, які представляють інші вихідні файли проекта чи логічні елементи. З допомогою команди **MegaWizard® Plug-In Manager** (меню **Tools**) можна створювати різні варіанти мега функцій та IP-ядер для включення їх в файл проекту.
3. Вказання початкових налаштувань проекту з допомогою редактора призначень (**Assignment Editor**), діалогового вікна **Settings** (меню **Assignments**), редактора топології (**Floorplan Editor**), та/або застосовуючи фіксовані логічні блоки (LogicLock™).
4. Створення проекту на системному рівні з допомогою генератора систем на кристалі (**SOPC Builder**) чи генератора систем цифрового опрацювання сигналів (**DSP Builder**).
5. Створення програмних файлів для процесорного ядра Nios® з допомогою редактора програмного забезпечення (**Software Builder**).

6. Синтез проекту з допомогою модуля аналізу та синтезу (**Analysis & Synthesis**).

7. Виконання функціонального моделювання проекту з допомогою симулятора (**Simulator**) та команди **Generate Functional Simulation Netlist**.

8. Виконання розміщення та трасування проекту з допомогою модуля трасування (**Fitter**).

9. Проведення попереднього аналізу споживаної потужності з допомогою програми **PowerPlay Power Analyzer**.

10. Проведення аналізу часових затримок проекту з допомогою програми аналізатора часових затримок (**Timing Analyzer**).

11. Виконання моделювання проекту з врахуванням часових затримок з допомогою симулятора.

12. Покращення часових характеристик проекту з допомогою повторного фізичного синтезу, використання фіксованих логічних блоків, налаштувань в діалоговому вікні **Settings** та в редакторі призначень.

13. Створення файлу для програмування мікросхеми з допомогою модуля асемблера (**Assembler**).

14. Програмування мікросхеми з допомогою утиліту програматора (**Programmer**) та обладнання **Altera**; чи перетворення формату файлу для програмування.

15. Відлагодження проекту з допомогою вбудованого логічного аналізатора (SignalTap® II Logic Analyzer), генератора контрольних точок (SignalProbe™).

Основною робочою одиницею в середовищі Quartus II є проект. Він створюється з допомогою спеціальної утиліти (New Project Wizard, меню File). При створенні нового проекту задається робоча директорія, назначається ім’я проекту та ім’я файлу верхнього рівня ієрархії. Додатково можна вказати вихідні файли проекту, користувальницькі бібліотеки, використовувані САПР інших фірм, вибране сімейство мікросхем (або використовувати автоматичний вибір сімейства компілятором Quartus II)

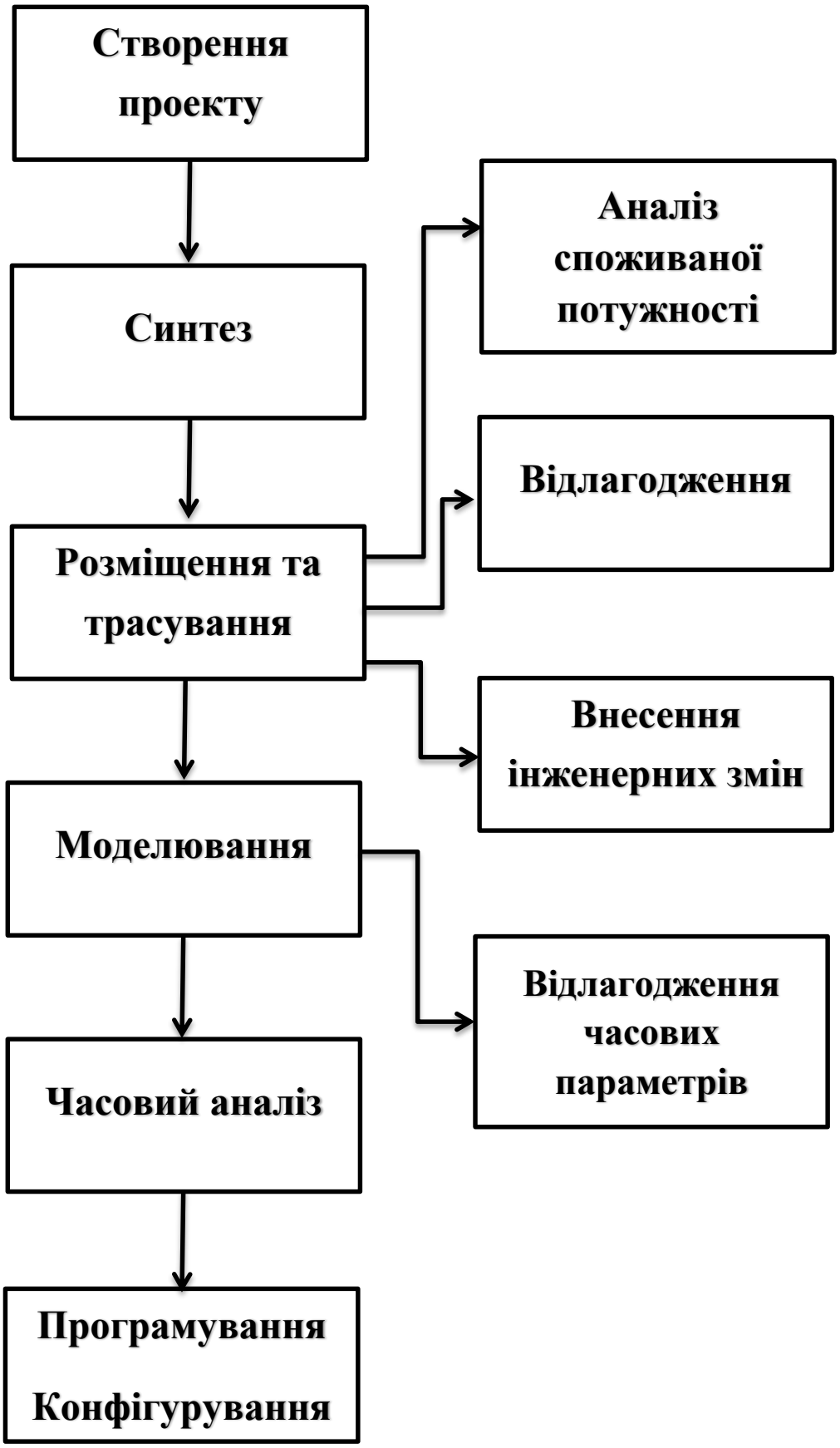


Рис.1. Процес проектування в Quartus II

Хід проектування

1. Створення нового проекту

1.1. Запустити САПР Quartus II.

1.2. В меню **File** вибирати **New Project Wizard…**. Відкриється нове вікно. Якщо на екрані з'являється сторінка **Introduction**, натисніть **Next.**

1.3. Створіть проект з допомогою утиліти **New Project Wizard** (рис. 2). На сторінці 1 потрібно задати наступну інформацію:

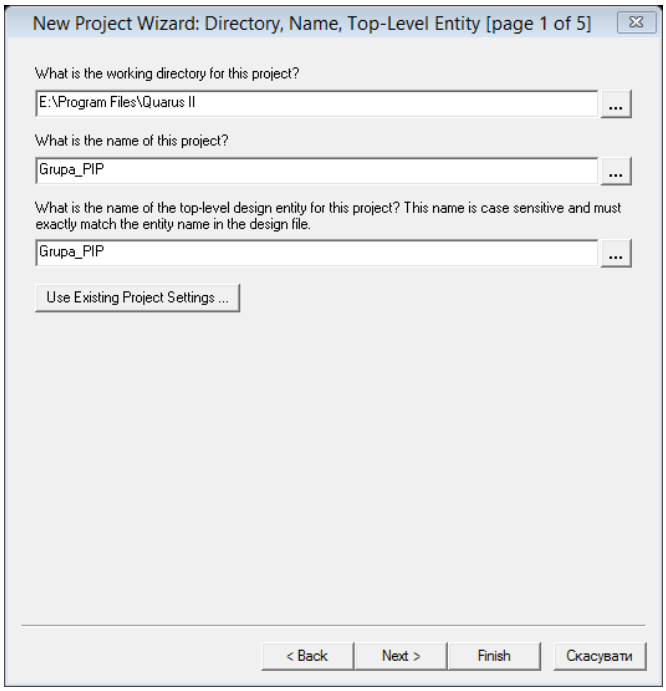


Рис. 2. Перша сторінка налаштувань New Project Wizard

1.4. Натисніть **Next** для того, щоб продовжити налаштування.

1.5. на сторінці 2 (рис. 3) майстер налаштувань пропонує вибрати файл верхнього рівня ієрархії. Оскільки готових файлів немає, то цей крок можна пропустити. Якщо ж такі файли є, тоді потрібно натиснути кнопку вибору та обрати файл верхнього рівня ієрархії з розширенням **.bdf**, він розташований в робочій директорії проекту. Натисніть **Open**, опісля **Add**, щоб додати файл у проект. В кінці натисніть **Next**

Дана дія не є обов'язковою, тому, що цей файл вже знаходиться в робочій директорії. Новий проект автоматично підключить його як складову частину проекту. Додавання файлів чи каталогів з файлами (бібліотеками) необхідно виконувати в тому випадку, якщо вони не знаходяться в робочій директорії проекту.

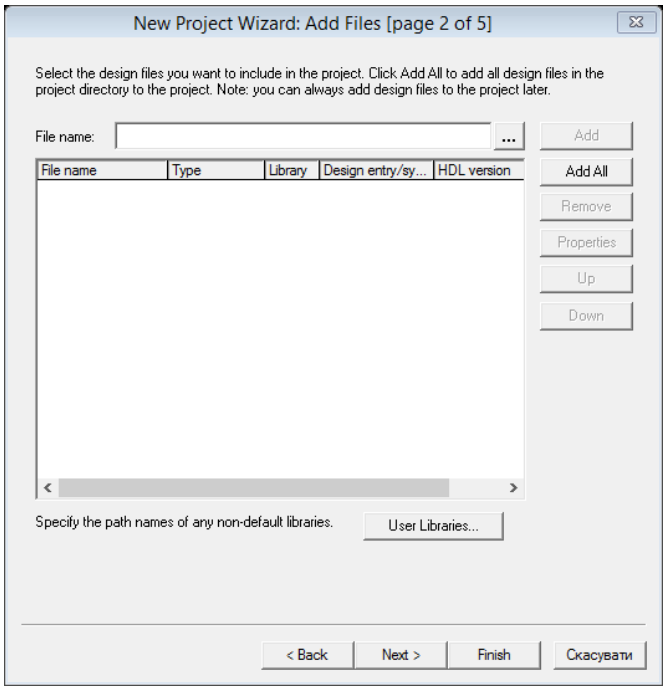


Рис. 3. Друга сторінка налаштувань New Project Wizard

1.6. На сторінці 3 обираємо сімейство **(Family) Cyclone II** (рис. 4).

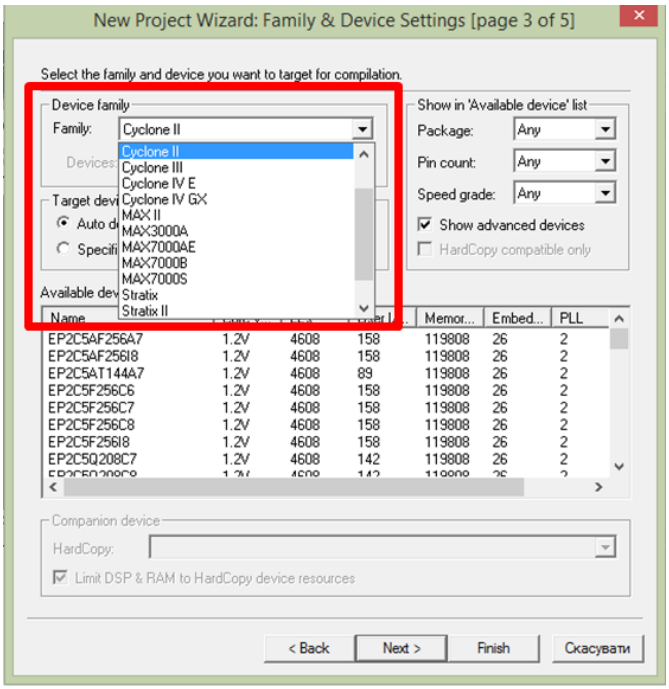


Рис. 4. Третя сторінка налаштувань New Project Wizard (вибір сімейства)

В правій частині вікна в розділі **Show in ‘Available device’ list** встановлюємо наступні значення: в графі **Package** обрати **FBGA**; **Pin count – 256**, **Speed grade – Fastest** (рис. 5).

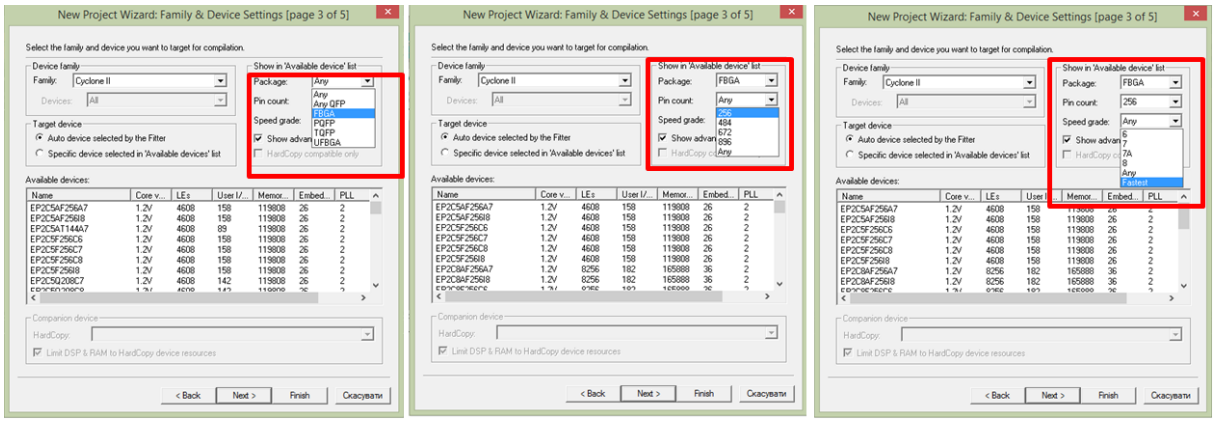


Рис. 5. Третя сторінка налаштувань New Project Wizard
(Show in ‘Available device’ list)

Ці налаштування обмежують список доступних мікросхем. У вікні **Available devices** обираємо мікросхему **EP2C5F256C6** (рис. 6), для продовження тиснемо **Next.**

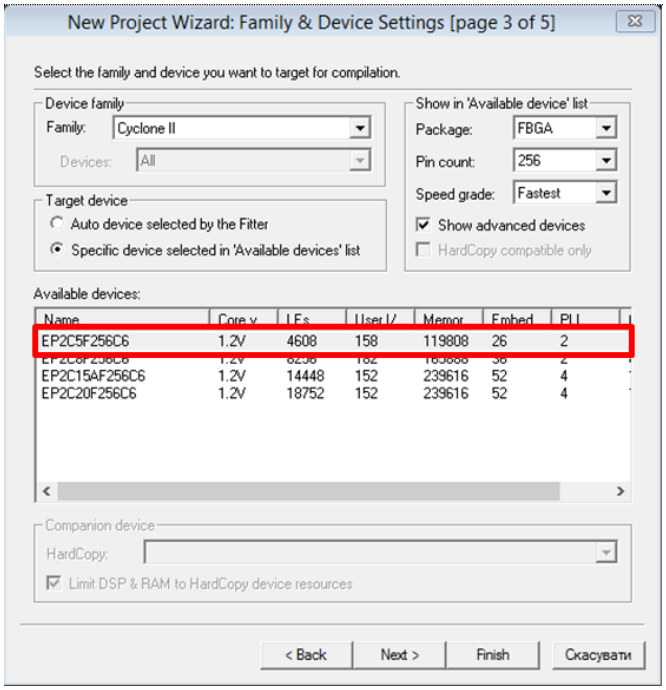


Рис. 6. Вибір мікросхеми для проекту

1.7. На сторінці 4 (рис. 7) можна вказати додатково використовувані САПР сторонніх виробників. В даній лабораторній роботі

використовується лише середовище **Quartus II**. Натисніть **Next** для того щоб пропустити цей крок.

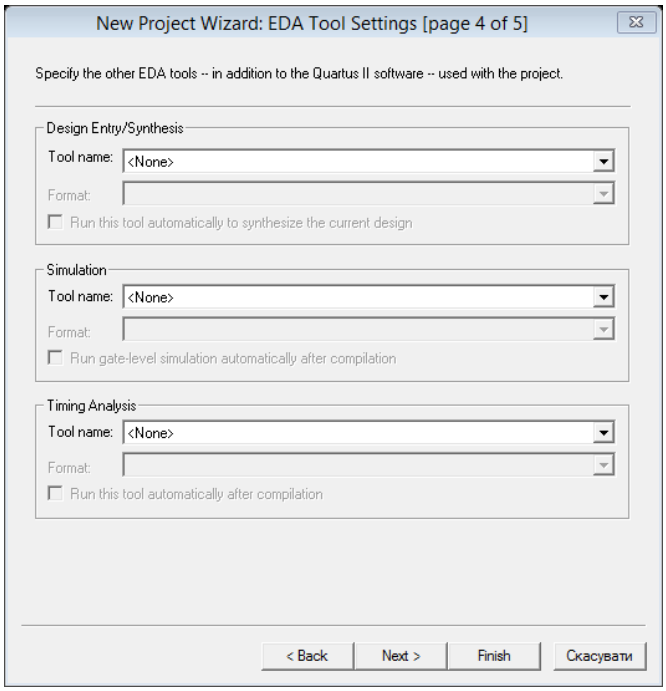


Рис. 7. Четверта сторінка налаштувань New Project Wizard

1.8. На рисунку 8 зображена кінцева сторінка. Для завершення створення проекту необхідно натиснути **Finish.**

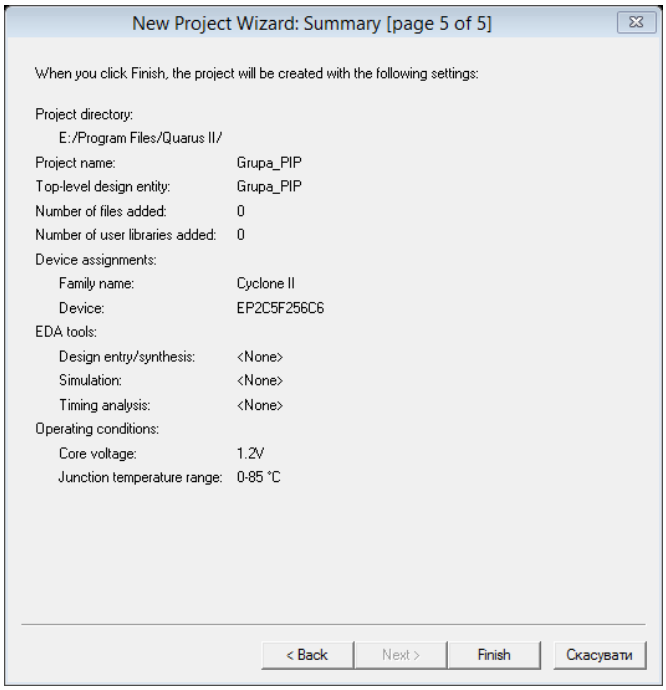


Рис. 8. Кінцева сторінка налаштувань New Project Wizard

Для продовження виконання завдання виходити із середовища **Quartus II** не потрібно. Закритий проект завжди можна відкрити з

допомогою команди **File –>Open Project**. Команда **File –>Open** дозволяє відкривати окремий файл (замість проекту), запобігаючи виконанню різних дій пов'язаних з опрацюванням проекту, наприклад, компіляцією.

На рисунку 9 представлено вікно створеного проекту у середовищі **Quartus II**.

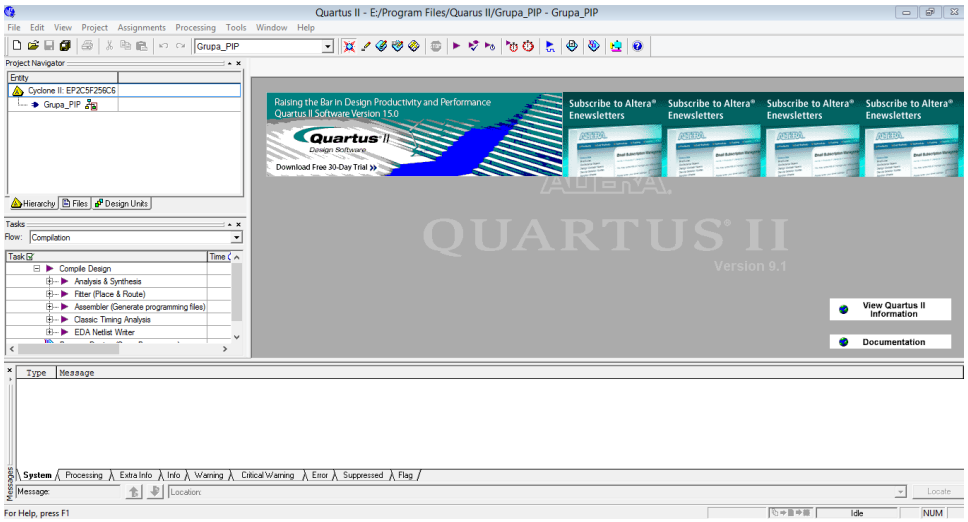


Рис. 9. Головне вікно нового проекту в середовищі **Quartus II**

2. Розробка конвеєрного помножувача

Для прикладу розглянемо схематичне представлення файлу верхнього рівня ієрархії, котрий необхідно реалізувати (рис. 10). Він складається із помножувача та блоку оперативної пам'яті RAM. Дані подаються на помножувач від зовнішнього джерела, а результат зберігається в блоці пам'яті, яким керує зовнішнє джерело. Тоді дані зчитуються із блоку оперативної пам'яті, використовуючи незалежну шину адреси для зчитування.

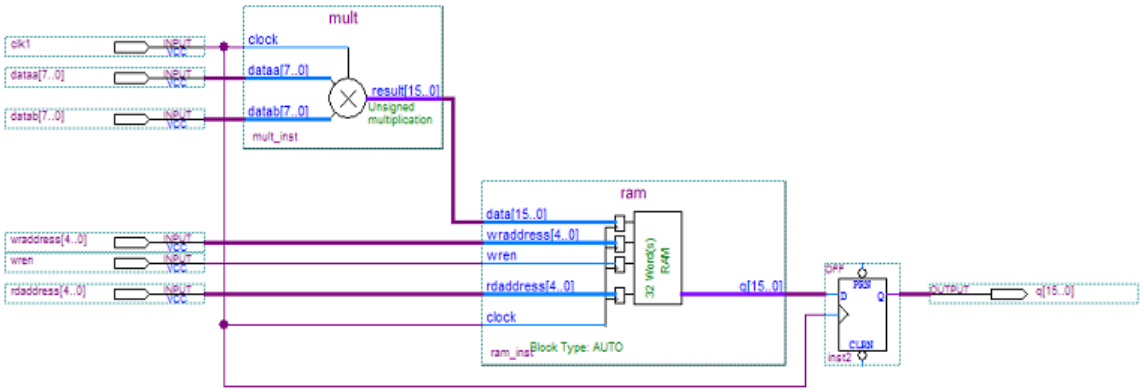
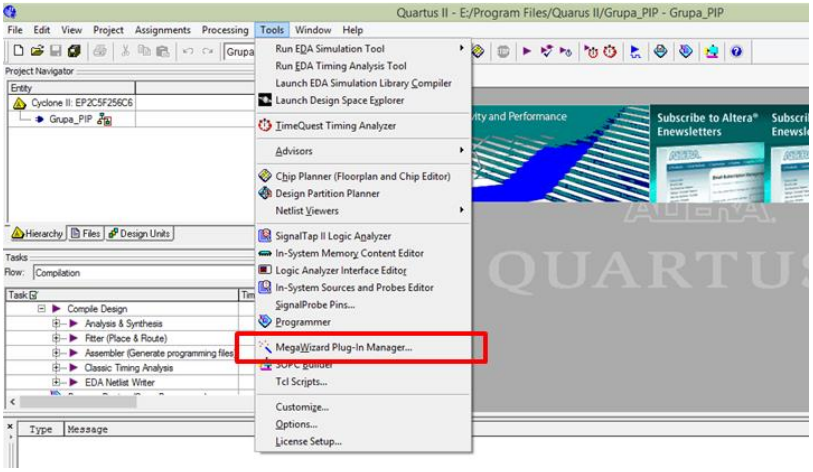


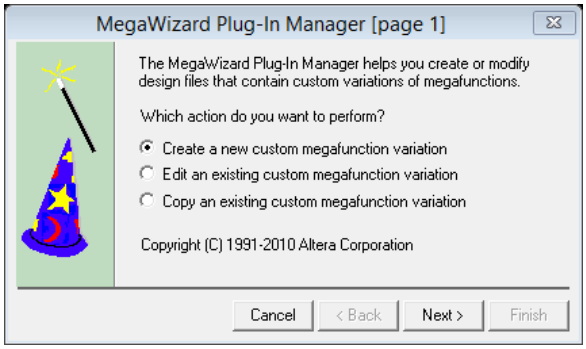
Рис. 10. Схематичне представлення файлу верхнього рівня ієрархії

2.1. Створення помножувача 8x8 з допомогою утиліти MegaWizard® Plug-in Manager

2.1.1. В меню **Tools** потрібно вибрати **MegaWizard Plug-In Manager**.



У відкритому вікні оберіть опцію **Create a new custom megafunction variation** і натисніть **Next**.



2.1.2. Оберіть необхідну мегафункцію. На сторінці 2 (рис. 11) відкрийте папку **Arithmetic** та оберіть **LPM_MULT**.

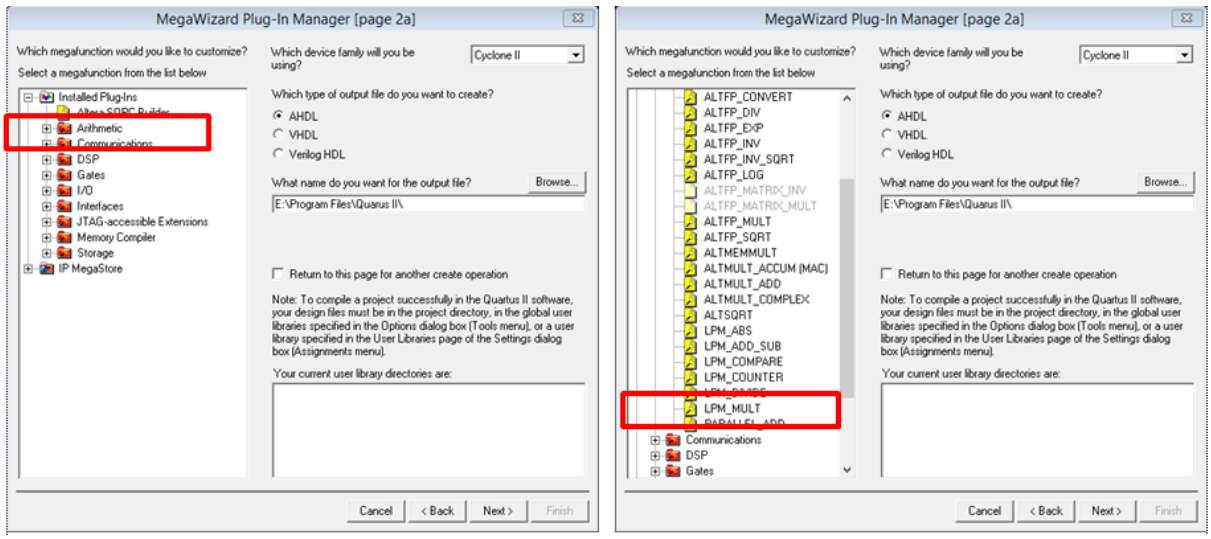
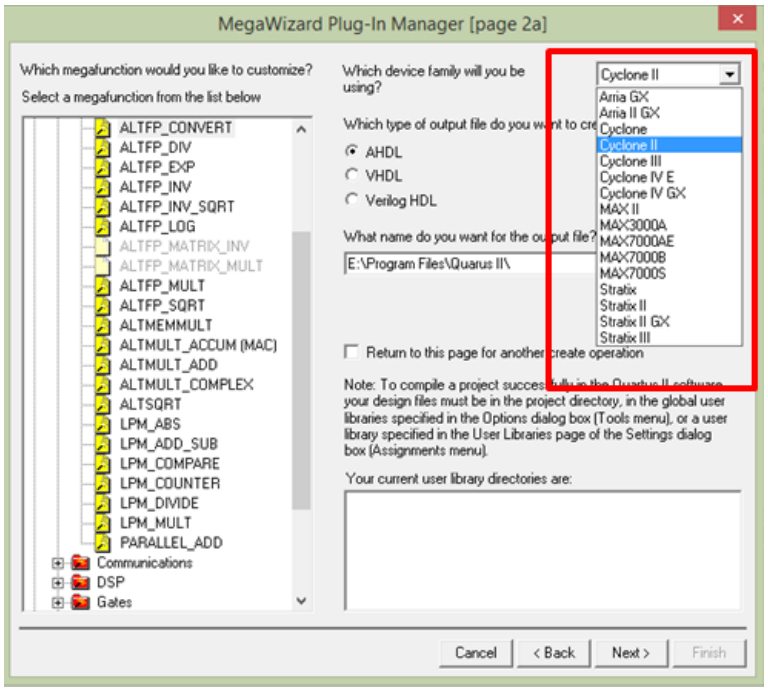


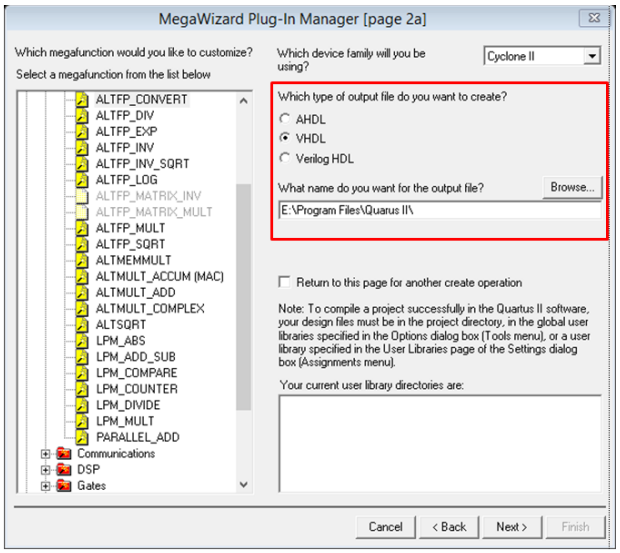
Рис. 11. Друга сторінка налаштувань MegaWizard Plug-In Manager

2.1.3. У контекстному меню з права вкажіть використовуване сімейство мікросхем **Cyclone II**.



Вибір сімейства мікросхем дозволяє утиліті **MegaWizard Plug-In Manager** визначити доступні ресурси сімейства та доступні мегафункції для утиліти. Можна створити мегафункцію для іншого сімейства мікросхем, при цьому необхідно обрати відповідне сімейство і створити для нього новий проект.

2.1.4. Оберіть мову описання інтерфейсу мегафункції (**VHDL** чи **Verilog HDL**) та вкажіть робочу директорію, де буде збережена мегафункція.



2.1.5. Задайте імя вихідного файлу – **mult**. Це можна зробити додавши його в кінці обраного шляху до потрібної директорії або не вказувати повний шлях для автоматичного збереження файлу в робочій директорії проекту. Натисніть **Next.**

2.1.6. На сторінці 3 **(General)** (рис. 12) вказується розрядність вхідних шин даних **dataa** та **datab** (вони можуть бути установлені автоматично). Натисніть **Next.**

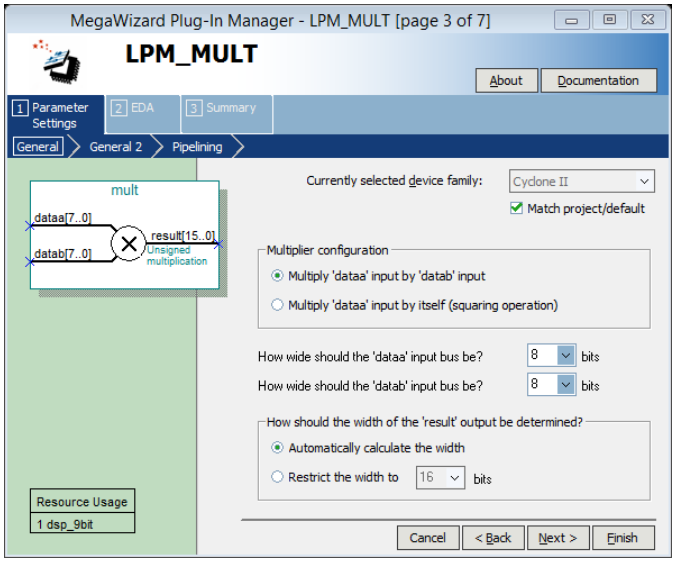


Рис. 12. Третя сторінка налаштувань MegaWizard Plug-In Manager

2.1.7. На сторінці 4 **(General 2)** (рис. 13) всі налаштування залишаються по замовчуванню (**datab** не є константою, використовується беззнакове множення та спосіб реалізації помножувача обирається по замовчуванню). Натисніть **Next.**

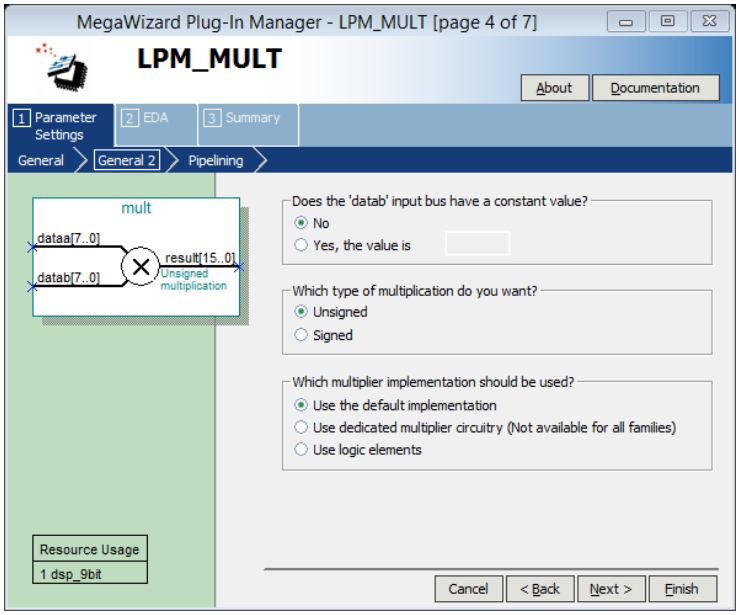


Рис. 13. Четверта сторінка налаштувань MegaWizard Plug-In Manager

2.1.8. На сторінці 5 **(Pipelining)** (рис. 14) вибираємо **Yes, I want an output latency of 2 clock cycles** (затримка видачі результату на 2 цикли тактової частоти). Натисніть **Next.**

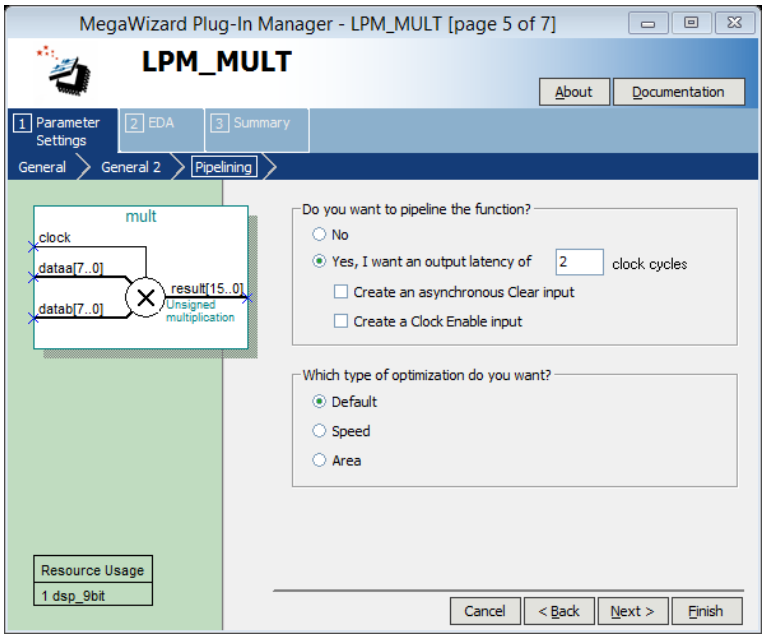


Рис. 14. П’ята сторінка налаштувань MegaWizard Plug-In Manager (вибір затримки видачі результату)

2.1.9. На наступній сторінці вказується файл **Lpm** (рис. 15), який використовується сторонніми САПР для моделювання мегафункції **LPM_MULT** (наприклад, в САПР ModelSim-Altera). В даному випадку для моделювання не застосовуються сторонні САПР. Натисніть **Next.**

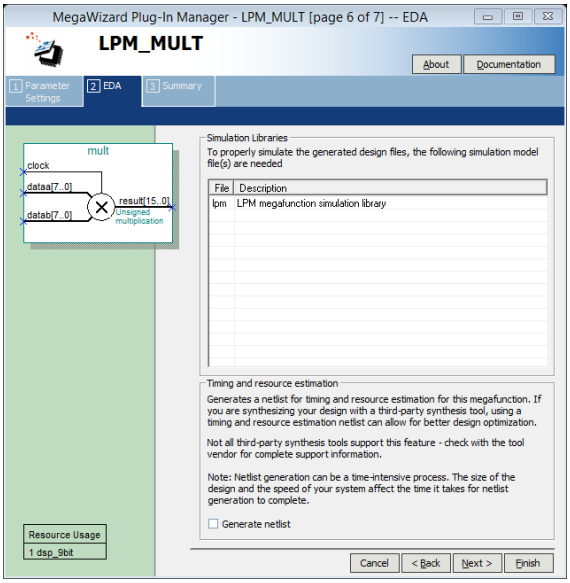


Рис. 15. Шоста сторінка налаштувань MegaWizard Plug-In Manager (вибір файлів для сторонніх САПР)

2.1.10. На сторінці 7 (рис. 16) потрібно відмітити створювані файли у наступному форматі: обираємо файл із розширенням .bsf та _inst.vhd.

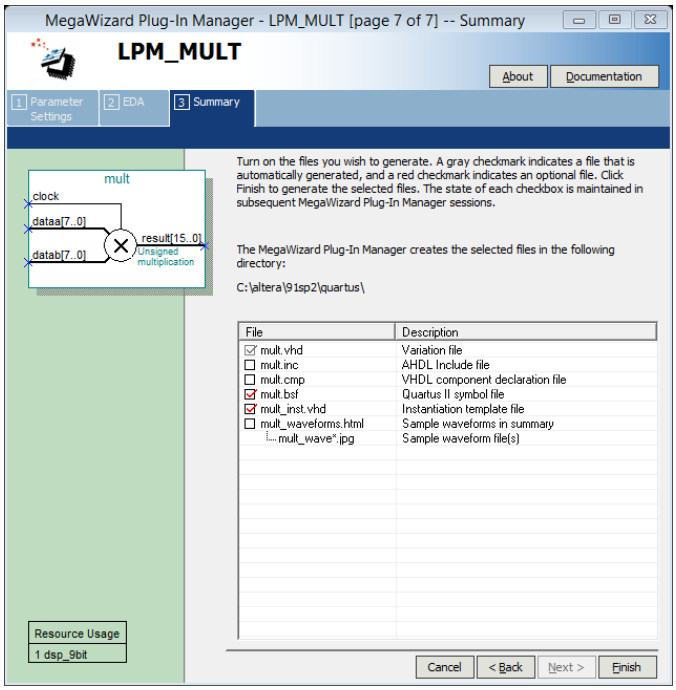


Рис. 16. Остання сторінка налаштувань MegaWizard Plug-In Manager

Натисніть **Finish** для завершення створення мегафункції. Помножувач створено. Якщо, з деяких причин, мегафункція створена не коректно чи щось було пропущено, заново відкрийте **MegaWizard Plug-In Manage** в меню **Tools**. Оберіть опцію редагування та виправте помилки. Натисніть **Finish**.

2.2. Створення 32×16 RAM з допомогою утиліти MegaWizard Plug-In Manager.

2.2.1. Знову відкрийте MegaWizard Plug-In Manager (**Tools -> MegaWizard Plug-In Manager**). Оберіть **Create a new custom megafunction variation** та натисніть **Next.**

2.2.2. Оберіть відповідну мегафункцію. На сторінці 2 розкрийте папку **Memory Compiler** та оберіть **RAM: 2-PORT** (рис. 17). Як і для реалізіції помножувача, оберіть сімейство мікросхем **Cyclone II** та мову описання мегафункцій – **VHDL** чи **Verilog**. В рядку імені вихідного файлу вкажіть **ram**. Натисніть **Next**.

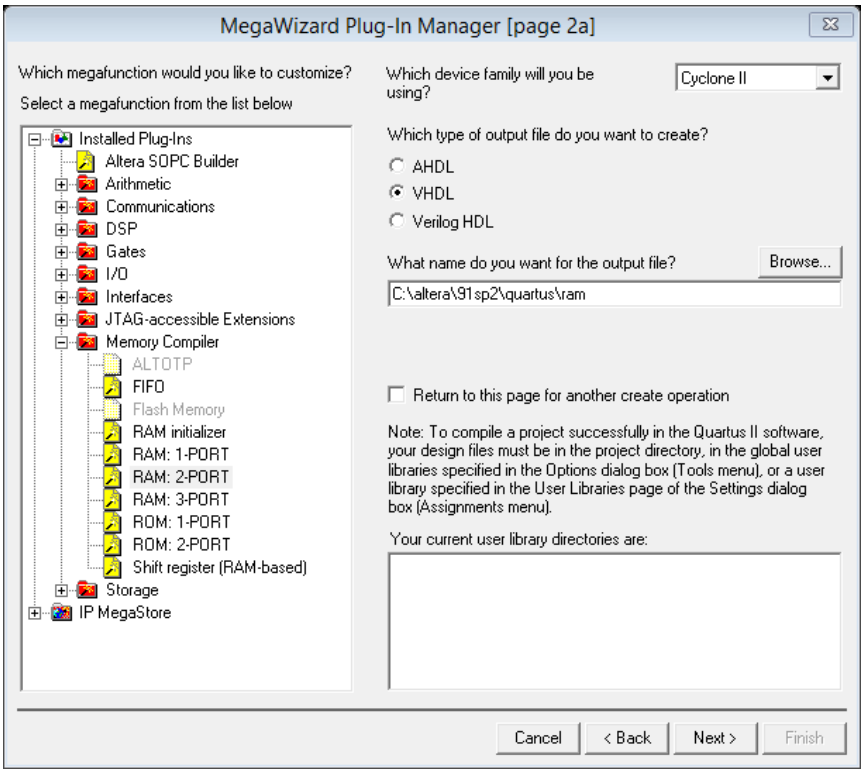


Рис. 17. Вікно налаштувань для 32×16 RAM

2.2.3. На сторінці 3 оберіть **With one read port and one write port** (рис. 18), в розділі вказання способу використання двох портів пам'яті. Решту налаштувань залишаємо по замовчуванню (розмір пам'яті визначається кількістю слів). Натисніть **Next.**

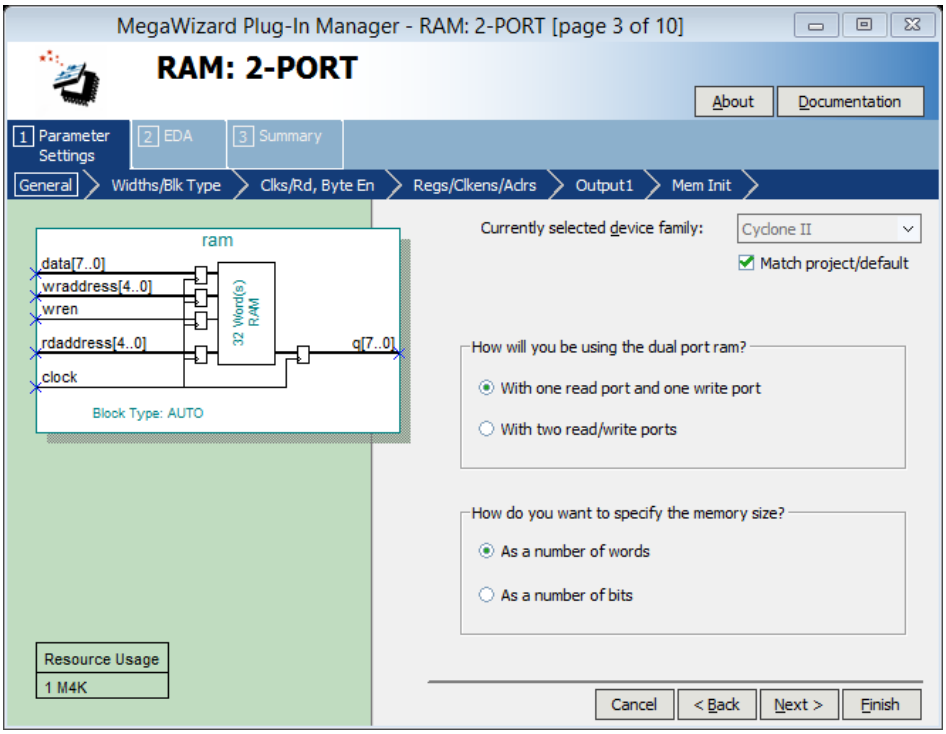


Рис. 18. Вікно вибору портів

2.2.4. На сторінці 4 **(Widths/Blk Type)** (рис. 19) встановлюється розрядність вхідного порту **data_a**, яка рівна 16 біт (в розділі **Read/Write Ports**). Тоді в обирається 32 – кількість 16-розрядних слів пам'яті (**16-bit words of memory**). Натисніть **Next.**

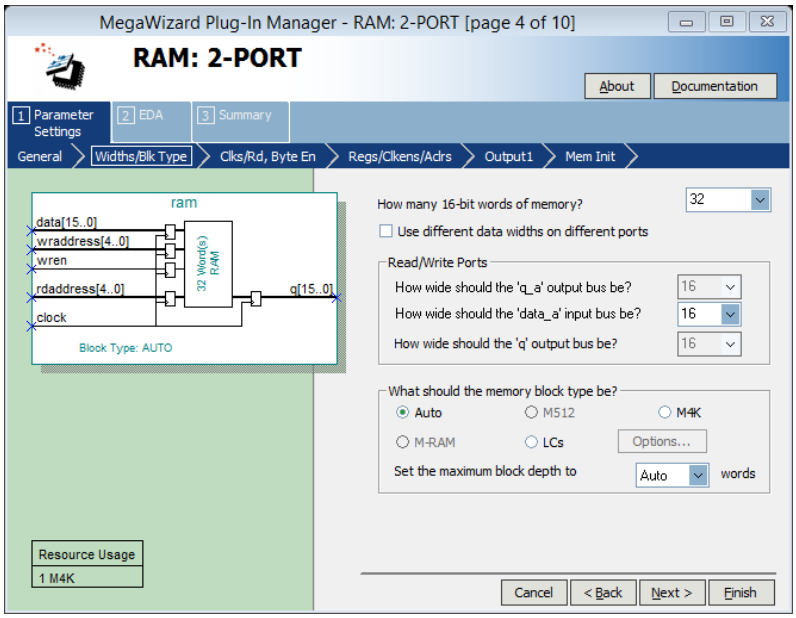


Рис. 19. Вікно вибору розрядності портів

2.2.5. П'ята сторінка залишається без змін. Натисніть **Next.**

2.2.6. На сторінці 6 (**Regs/Clkens/Aclrs**) (рис. 20), відключається опція **Read output port(s) 'q'**, щоб відключити встановлення вихідних регістрів для порту **q**. Інші налаштування залишаються без змін. Сьома сторінка залишається без змін. Натисніть **Next** двічі.

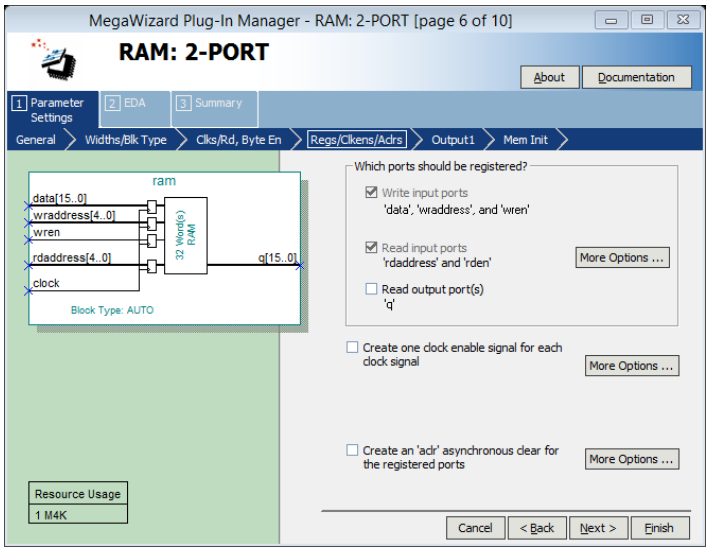


Рис. 20. Шоста сторінка налаштувань

2.2.7. На 8 сторінці (**Mem Init**) (рис. 21) оберіть **Yes**, щоб вказати файл ініціалізації блоку пам'яті. Після того, як поле стало доступним, вписується ім'я файлу: **ram.hex**. Цей файл буде створено пізніше, але вказати його ім'я можна і на даному етапі. Натисніть **Next.**

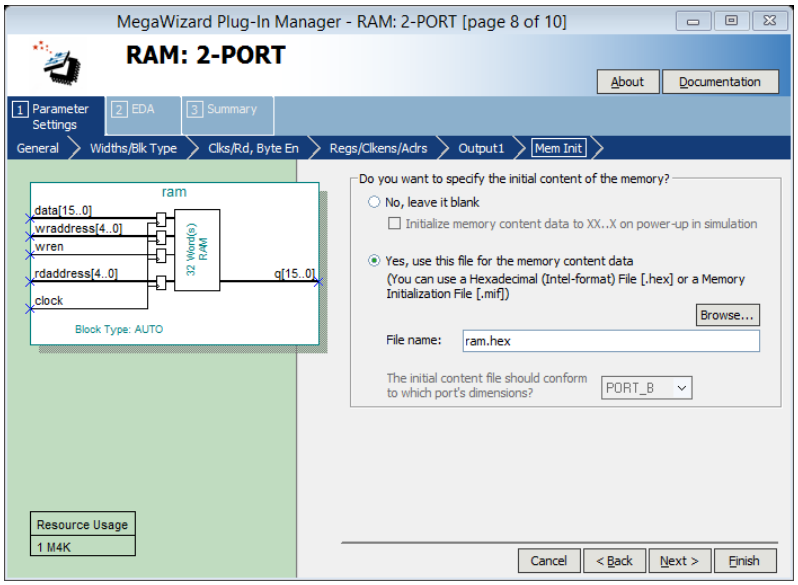


Рис. 21. Восьма сторінка налаштувань

2.2.8. На наступній сторінці відображається файл **altera_mf**, який необхідний для моделювання цієї мегафункції з допомогою сторонніх САПР. Натисніть **Next**. На кінцевому етапі (рис. 22) обираються файли для створення мегафункції **ram**, котрі були обрані для мегафункції у пункті 2.1.10. Натисніть **Finish.**

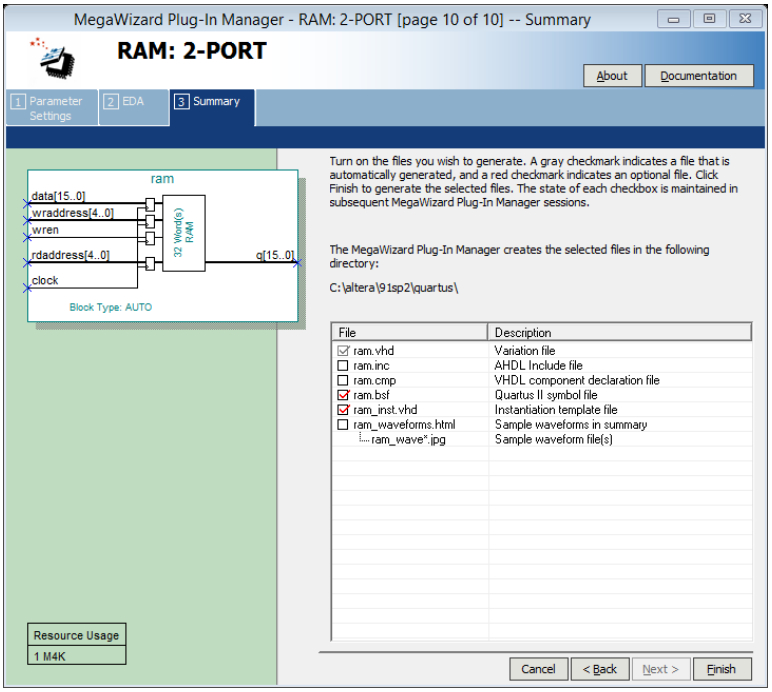


Рис. 22. Кінцевий етап створення мегафункції ram.

На даному етапі вже створено 2 компоненти, необхідних для даного проекту. Далі необхідно створити **HEX** фал, котрий описує вміст для ініціалізації оперативної памяті **RAM**.

2.3. Створення HEX файлу з допомогою редактора Memory Editor

2.3.1. В меню **File** обирається команда **New** (рис. 23). Обирається **Memory Files > Hexadecimal (Intel Format) File**. Натисніть **Ok**.

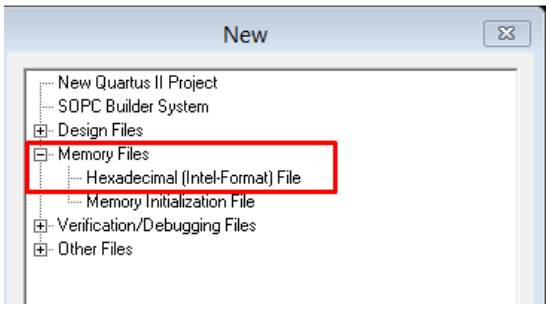
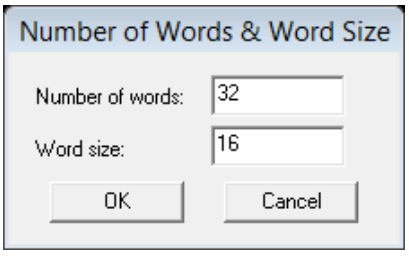


Рис. 23. Створення HEX файлу.

2.3.2. У діалоговому вікні налаштувань розміру області пам'яті встановлюються наступні параметри: кількість слів – **number of words – 32**, розмір слова – **word size – 16**. Натисніть **Ok.**



Вікно редактора пам'яті **Memory Editor** відображає заданий простір пам'яті. Якщо простір памяті відображається не так, як показано на рис. 24, можна змінити кількість стовпців (меню **View**) на 16 та формат значень у **Hexadecimal.**

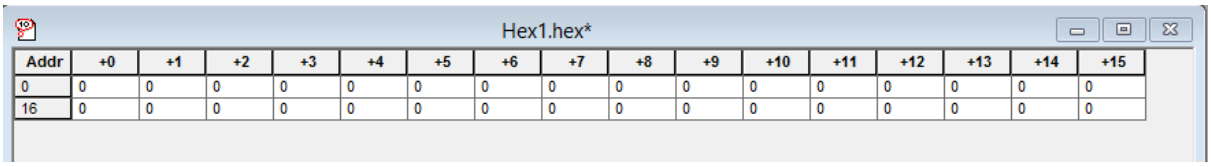


Рис. 24. Створення Hex файлу.

- З допомогою діалогового вікна **Custom Fill Cells** (рис. 25), можна ввести значення для ініціалізації області пам'яті. Можна обрати одну із наступних дій:

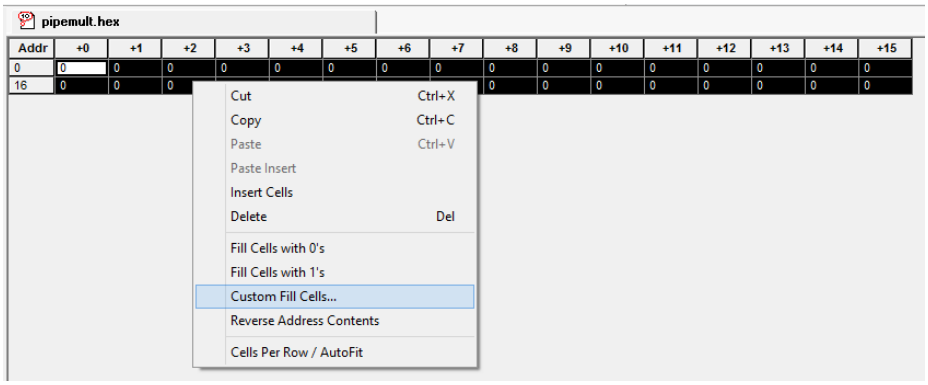


Рис. 25. Вибір діалогового вікна Custom Fill Cells

• Повторювана послідовність: вводиться послідовність значень, котрі будуть повторюватись в пам'яті, відділяючи їх пропуском чи комою.

• Зростаюча/зменшувана послідовність: вводиться початкове значення та значення величини, на яку буде змінюватись кожне наступне значення (збільшуватись чи зменшуватись) (рис. 26).

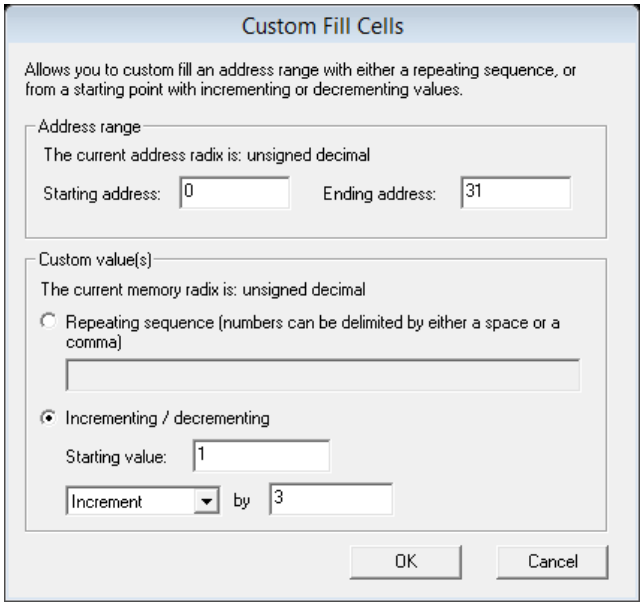


Рис. 26. Приклад вибору значень для ініціалізації області пам'яті

- Збережіть файл як **ram.hex** та закрийте його.

2.4 Додавання блоків у проект та створення необхідних зв'язків

2.4.1. Щоб завершити проект необхідно створені блоки поєднати у схему. Для цього створюється діаграмний файл: File/New… (рис. 27) та зберігається з ім'ям **pipemult.bdf.**

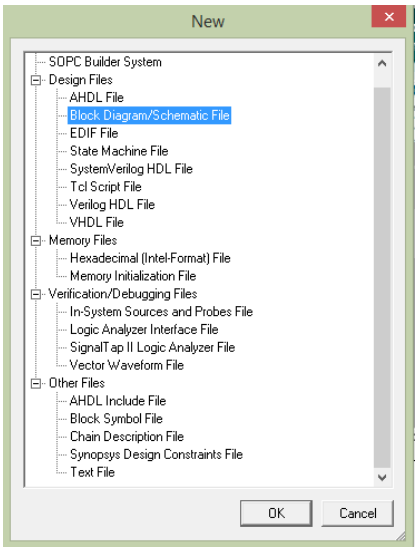


Рис. 27 Створення діаграмного файлу.

2.4.2. Щоб додати створені блоки, двічі клацніть лівою клавішою миші у схемотехнічному вікні. У вікні **Symbol** розкрийте папку **Project**, щоб додати необхідні блоки (**mult**), тоді натисніть ліву клавішу миші в полі креслення, щоб вставити вибраний елемент у вказане місце.

Зауваження: три вхідних порти в лівій частині помножувача повинні співпадати із входами, які присутні на схемі. Якщо це не так, можливо, мегафункція була створена неправильно. В такому випадку натисніть **Esc**, щоб відмінити вставку символу, заново відкрийте утиліту **MegaWizard Plug-In Manager** та оберіть опцію **Edit an existing megafunction variation**. Вкажіть елемент та виправте помилки та повторіть установку символу.

2.4.3. Правою клавішею миші натисніть на блок **mult** та оберіть команду **Properties**, в рядку **Instance name** змініть ім’я **inst** на **mult_inst**. Натисніть **OK.**

2.4.4. Знову відкрийте вікно **Symbol**. Оберіть елемент **ram** та встановіть його у вказане місце.

Зауваження: 4 нижніх входи блоку **ram** повинні співпадати із вхідними портами. Вхід **data** має залишитись не підключеним.

2.4.5. Правою клавішею миші натисніть на блок **ram** та оберіть команду **Properties**, в рядку **Instance name** змініть ім’я **inst** на **ram_inst**. Натисніть **OK.**

2.4.6. Відкрийте вікно **Symbol** та в рядку **Name** введіть **output**. Елемент має визначитись автоматично. Натисніть **OK** та встановіть

елемент біля вихідного контакту регістра **inst2**. Двічі клацніть в області вказання імені компонента та замініть **pin_name** на **q[15..0]**.

2.4.7. З допомогою панелі інструментів з'єднайте між собою вихід елементу **mult (result)** та вільний вхід елементу **ram (data)** (рис. 28). Збережіть кінцевий варіант проекту.

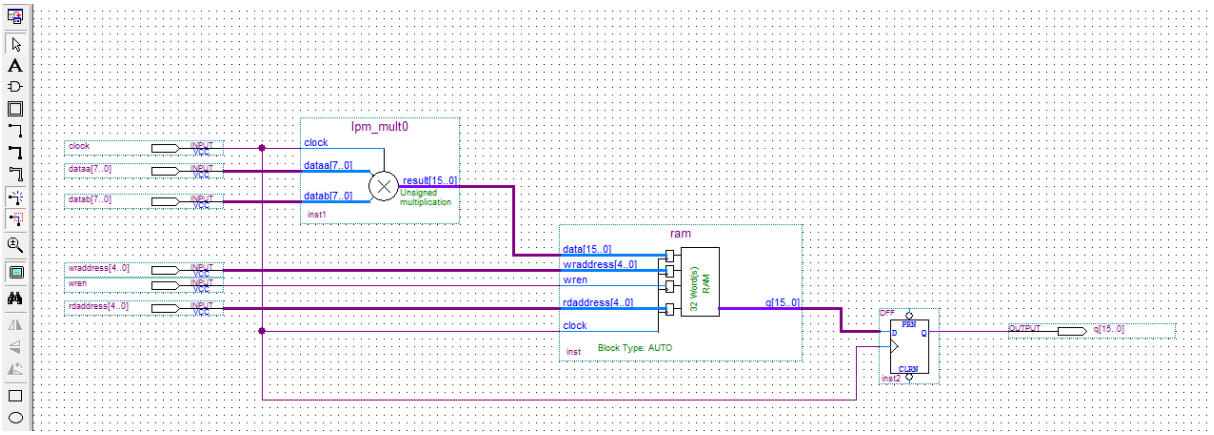


Рис. 28. Схемотехнічне вікно із готовою схемою

2.5. Зберігання та перевірка проекту

2.5.1. В меню **Processing** оберіть команду **Start / Start Analysis & Elaboration** (рис. 29). Ця команда виконує перевірку наявності всіх файлів у проекті та правильність їх під'єднання.

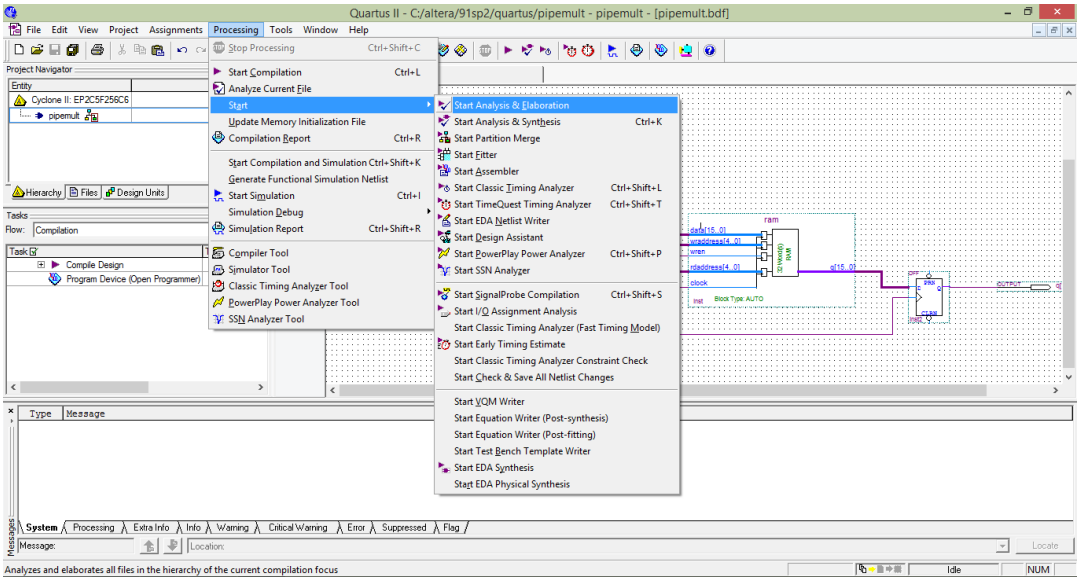


Рис. 29. Перевірка справності проекту

2.5.2. Коли аналіз завершиться, натисніть **OK**. Якщо були виявлені помилки, перевірте всі зв'язки чи повторно викличте **MegaWizard Plug-In Manager** для виправлення помилок у мегафункціях. (рис. 30)

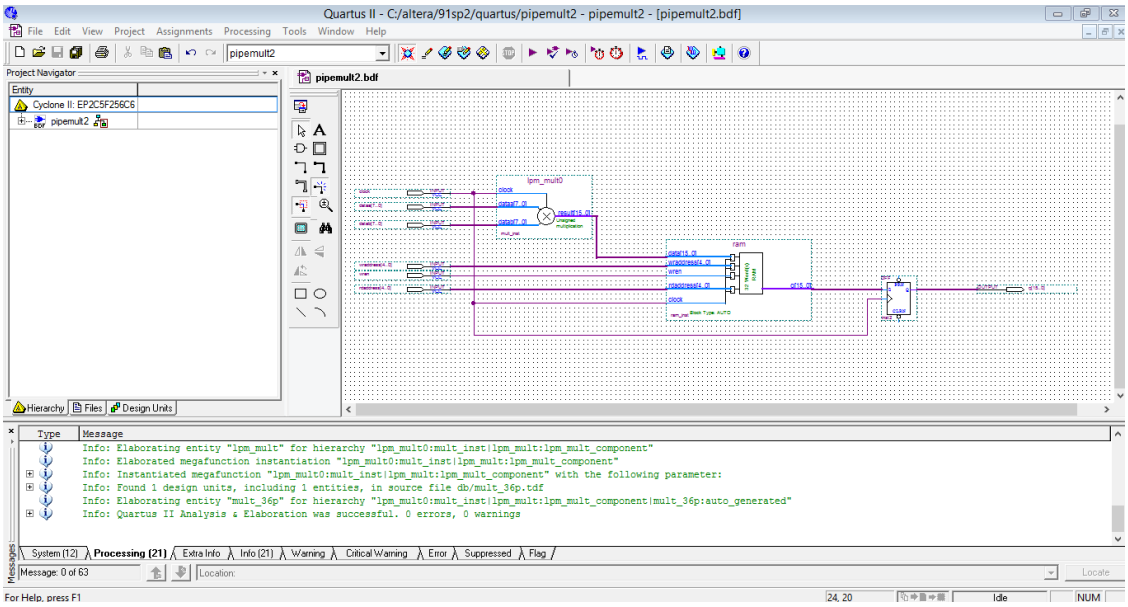


Рис. 30. Вигляд вікна проекту після завершення перевірки.

Завдання на лабораторну роботу 6

1. Ознайомитись із теоретичними відомостями.
2. Ознайомитись із роботою середовища Quartus II
3. Створити новий проект у середовищі Quartus II назвавши його «Група_прізвище» (наприклад: СІ-42_Коваль).
4. Створити та відредагувати схему електричну принципову відповідно до заданого варіанту.
5. Зробити висновки. Результати у вигляді скріншотів представити у звіті.

Контрольні питання

1.
2.
3.
4.
5.

Лабораторна робота № 7

Моделювання проекту в середовищі Quartus II

Тема роботи: навчитись налаштовувати режим моделювання, створювати тестові вхідні сигнали і виконувати аналіз результатів моделювання

Теоретичні відомості

Перевірку функціонування проекту (функціональне і тимчасове моделювання) можна проводити з використанням САПР сторонніх фірм або вбудованого в **Quartus II** симулятора.

Quartus II надає наступні можливості для проведення моделювання за допомогою САПР сторонніх фірм:

- інтерфейс зв'язку **Native Link** з іншими САПР;
- генерація вихідного файлу зв'язків;
- бібліотеки для функціонального і тимчасового моделювання;
- оцінка споживаної потужності утилітою **Power Gauge**;
- генерація тестового файлу та файлу ініціалізації пам'яті.

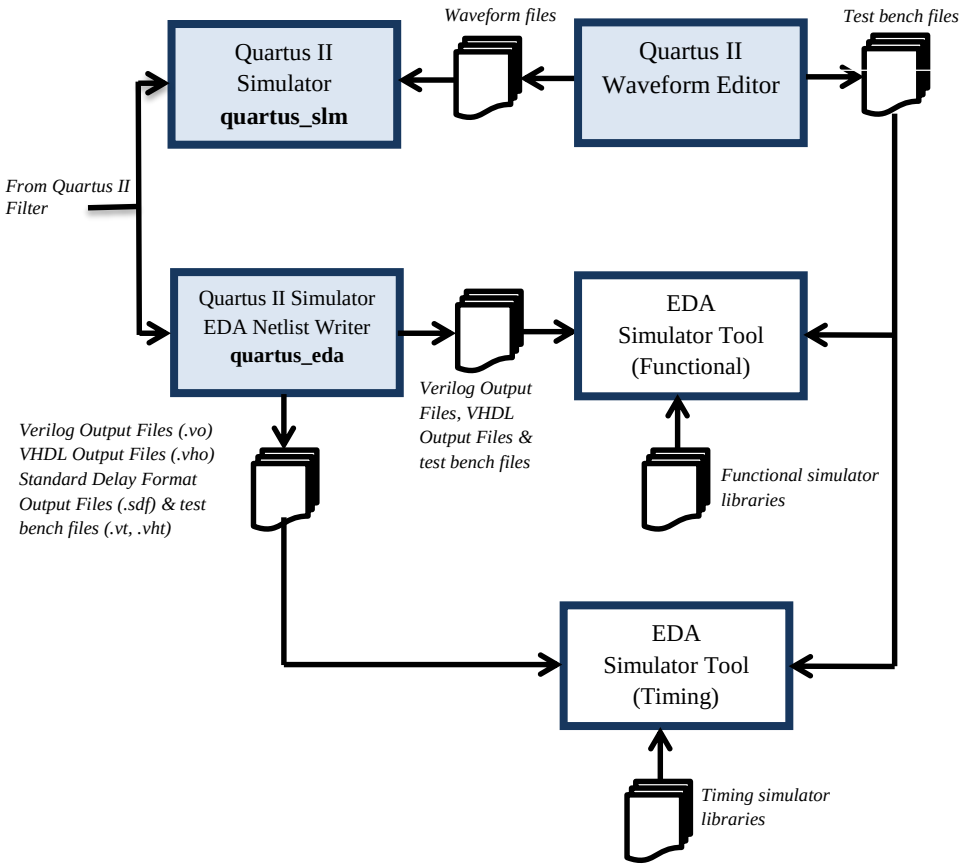


Рисунок 31. Основні етапи моделювання в середовищі Quartus II

Хід проектування

1. Налаштування моделюючої програми

1.1. Відкрийте проект **pipemult.qpf** з робочої директорії.

1.2. У меню **Assignments** виберіть команду **Settings**. У категорії **Simulator Settings** встановіть режим **Functional** у випадаючому меню **Simulation mode**.

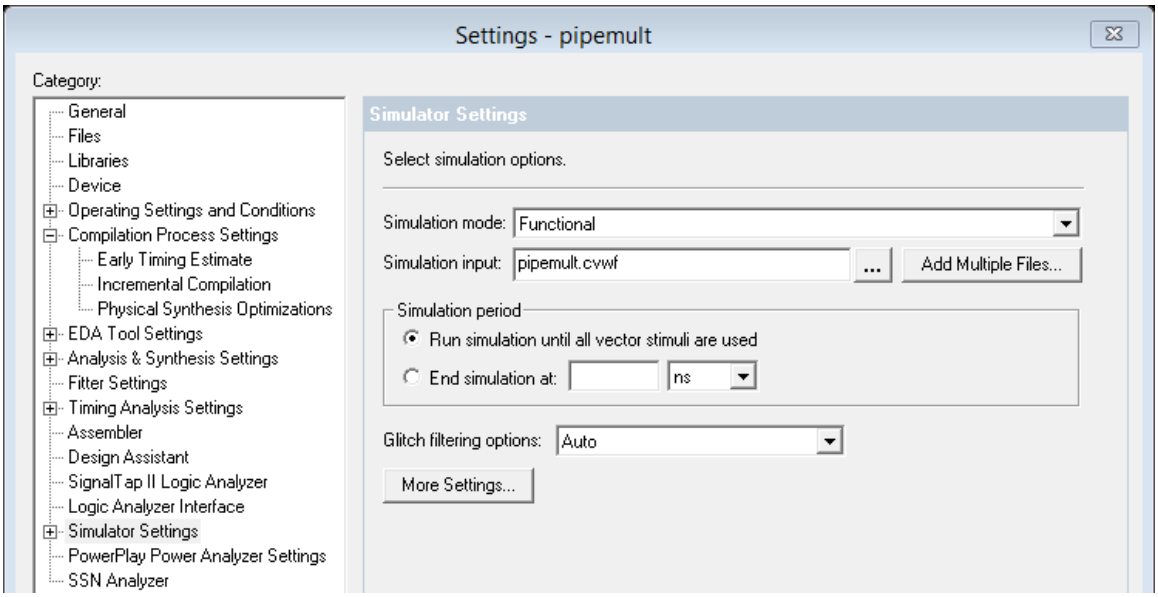
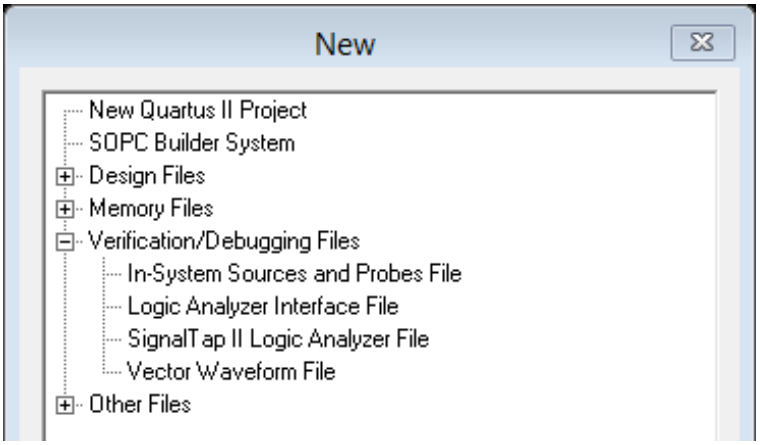


Рисунок 32. Вікно налаштувань проекту

В якості вихідного файлу для моделювання вкажіть файл із розширенням **.cvwf**. Натисніть **OK**.

2. Створення vwf файлу.

2.1. У меню **File** виберіть команду **New**. У діалоговому вікні **New** розкрийте закладку **Verification/Debugging Files**, виберіть **Vector Waveform File** і натисніть **OK**.



3. Формування модельованих сигналів

3.1. У меню **Edit** виберіть підменю **Insert**, а в ньому команду **Insert Node or Bus**. У діалоговому вікні **Insert Node or Bus** натисніть кнопку **Node Finder**.

3.2. У вікні **Node Finder** встановіть режим **Pins: all** у випадаючому меню віконця **Filter**. Натисніть кнопку **List**.

3.3. Виділіть сигнали **clk1**, **dataa**, **datab**, **wraddress**, **rdaddress**, **wren**, і **q**. Натисніть кнопку > для перенесення виділених сигналів у вікно **Selected Nodes**. Натисніть **OK**. Натисніть ще раз **OK** в діалоговому вікні **Insert Node or Bus**.

3.4. У файлі **.vwf** виділіть сигнали **dataa**, **datab**, **wraddress**, **rdaddress** і **q**. Натисніть праву кнопку миші і виберіть команду **Properties**. Змініть формат представлення сигналів із **Binary** на **Hexadecimal**. Натисніть **OK**.

3.5. У меню **Edit** виберіть команду **End Time**. У діалоговому вікні **End Time** встановите значення параметра **Time** рівне **100 ns**. Зверніть увагу на правильність установки даного параметра. Натисніть **OK**.

3.6. Викличте команду зміни масштабу **Zoom** (кнопка ). Натисніть праву кнопку миші в будь-якому місці графічного поля (повторюйте дію до тих пір, поки в даному полі не буде розміщатися весь інтервал моделювання, рівний **100 ns**). Вимкніть режим масштабування.

4. Створення стимулюючих сигналів

4.1. У файлі **.vcf** повинні відображатися всі обрані Вами сигнали. Задайте зміну сигналів **clock**, **dataa**, **datab**, **wraddress**, **rdaddress** і **wren** так, як показано на рисунку нижче.

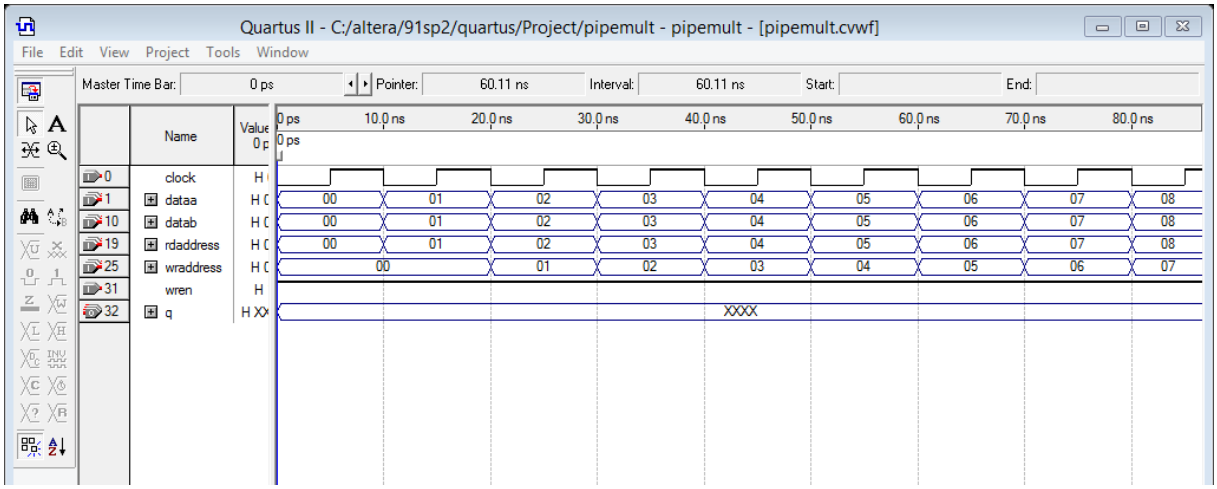


Рисунок 33. Зображення заданих сигналів проекту

Для зміни сигналу, виділіть його або виділіть окремий часовий інтервал для нього і встановіть відповідне значення за допомогою команд в панелі інструментів або меню **Edit**. Можна використовувати команди **Insert Clock** і **Insert Count Value** для спрощення процедури зміни значень сигналу.

5. Збереження файлу і запуск моделюючої програми

5.1. Збережіть тестовий файл під ім'ям проекту з розширенням **.cvwf** (**compressed vector waveform file**).

5.2. У меню **Processing** виберіть команду **Start Simulation**.

5.3. Якщо з'явилось повідомлення про помилку необхідно у меню **Processing** виберати команду **Generate Functional Simulation Netlist**.

4. Знову виберіть команду **Start Simulation**. Після закінчення моделювання з'явиться повідомлення "**Simulation was Successful**". Натисніть **OK**.

6. Аналіз результатів

6.1. Вікно **Simulation Report** відкриється автоматично при запуску моделюючої програми.

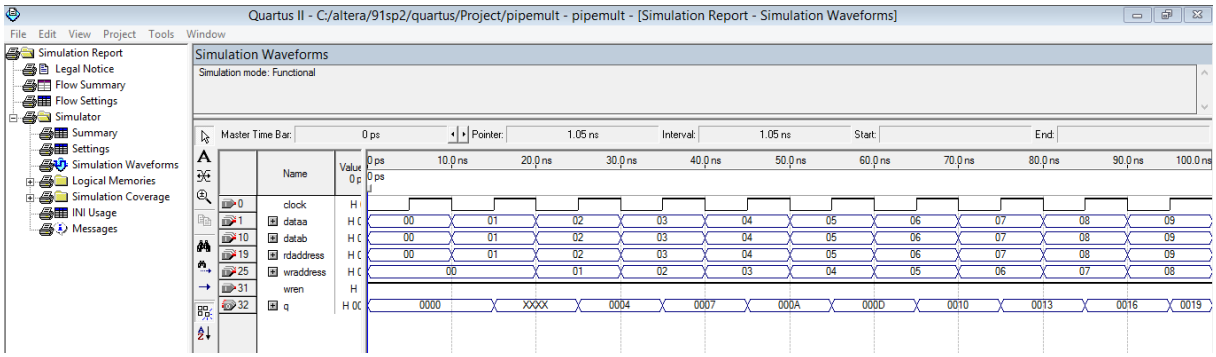


Рисунок 34. Зображення сигналів проекту після моделювання

6.2. У вікні **Simulation Waveform** звіту моделюючої програми повинні відображатися графічні результати моделювання, як показано на рисунку 34.

Завдання на лабораторну роботу 7

1. Ознайомитись із теоретичними відомостями.
2. Розробити власний пристрій з використанням арифметичних функцій і блоків пам'яті.
3. Створити для нього проект. Виконати моделювання.

4. Зробити висновки. Результати у вигляді скріншотів представити у звіті.

Контрольні питання

1.
2.
3.
4.
5.

Лабораторна робота № 8

Компіляція проекту в середовищі Quartus II. Аналіз результатів компіляції

Мета роботи: навчитись використовувати інструменти **Compilation Report**, **RTL Viewer**, **Technology Map Viewer** та **Chip Planner** для аналізу результатів компіляції проекту в середовищі **Quartus II**.

Теоретичні відомості

Процес повної компіляції проекту в середовищі Quartus II включає послідовне виконання наступних етапів: синтаксичний аналіз та синтез, розміщення і трасування, формування конфігураційного файлу, часовий аналіз. Всі ці етапи представлені у вікні завдань менеджера проекту (рис. 35).

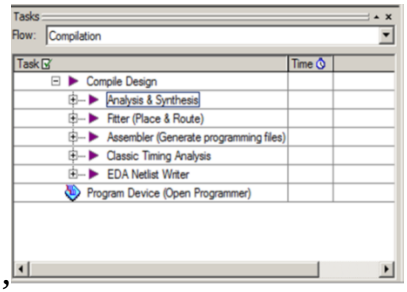


Рисунок 35. Складові етапи компіляції проекту

В свою чергу, окремі етапи процесу компіляції проекту можуть містити ряд вкладених самостійних процесів. Щоб їх побачити, потрібно клацнути лівою клавішею миші по значку «+» поруч з назвою етапу. На рисунку 36, наведено вікно задач, в якому представлено всі процес, які входять в етап аналізу та синтезу проекту.

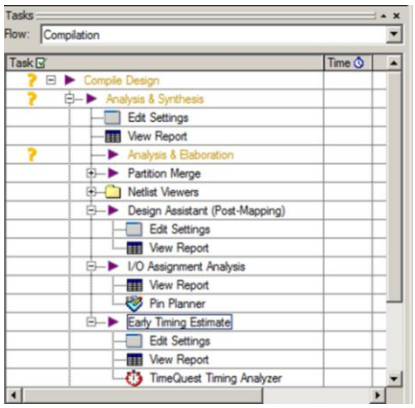


Рисунок 36. Процеси, що запускаються при виконанні аналізу та синтезу проекту

Як видно, цей етап включає процес аналізу призначень контактів вводу-виводу, процес виконання призначень помічником проекту (**Design Assistant**), процес ранньої оцінки часових характеристик проекту та ін.

Ті процеси, котрі містять поруч з назвою знак ▶, можуть бути виконані шляхом подвійного клацання миші по їх назві. Однак, в деяких випадках, їх виконання потребує виконання попередніх процесів компіляції. У вікні задач містяться рядки **Edit Settings** (див. рис. 36), подвійне клацання миші по котрих приводить до появи на головній панелі менеджера проекту вікна з установками для відповідних процедур. Користувач може виконати редагування установок, використовуючи ці вікна.

На рисунку 37 наведено приклад вікна з установками для етапу аналізу та синтезу проекту. В якості критерію оптимізації проекту вибрано швидкодію проекту. Подвійне клацання клавішою миші на **View Report** приводить до появи на головній панелі менеджера проекту вікна із звітом про виконання відповідної частини процесу компіляції.

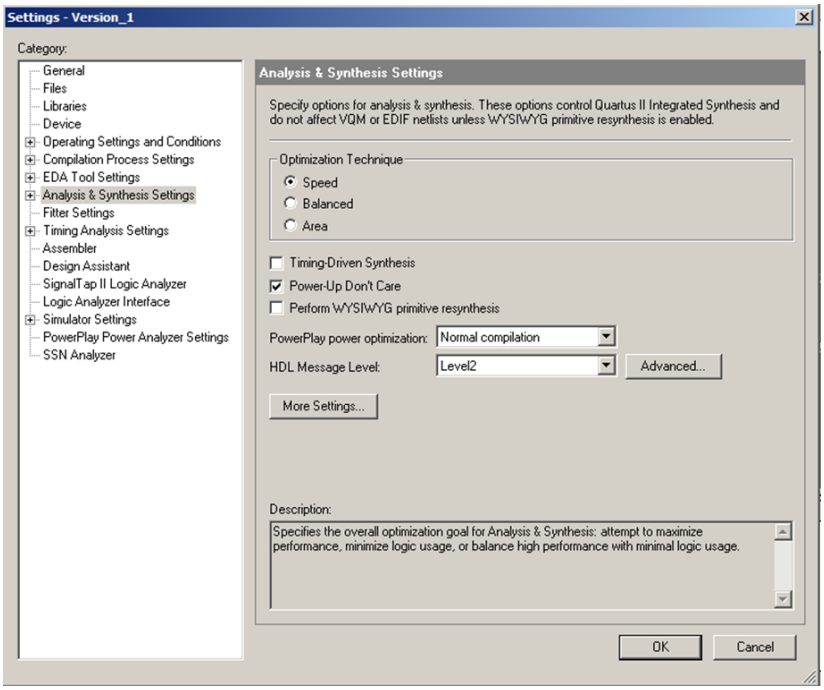


Рисунок 37. Вікно налаштувань для етапу аналізу та синтезу проекту.

Для налаштувань параметрів кожного етапу компіляції проекту також може бути використана піктограма  на панелі інструментів чи команда *Settings*… із меню **Assignments** менеджера проекту.

Виконати можна як повну компіляцію всього проекту, так і окремих складових етапів. Для запуску процесу повної компіляції проекту використовується піктограма  на панелі інструментів чи команда *Start Compilation* із меню **Processing** менеджера проекту.

Для запуску процесу виконання окремих етапів чи кроків компіляції використовується команда *Start* із меню **Processing** менеджера проекту. При наведенні курсору миші на цю стрічку з'являється ще одне контекстне меню (рис. 38), в якому потрібно вказати виконуваний етап компіляції.

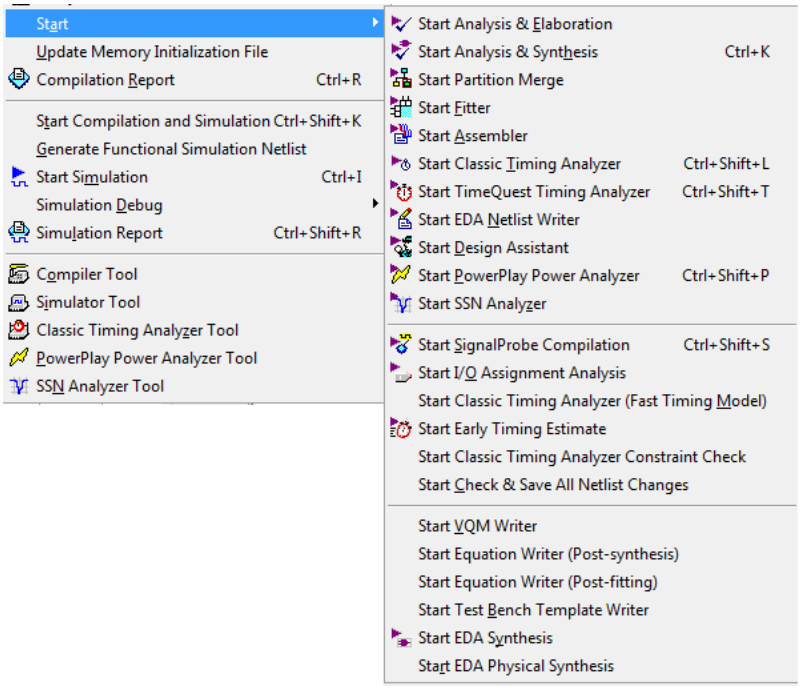


Рисунок 38. Фрагмент меню **Processing** менеджера проекту та контекстне меню команди *Start*

Для індикації процесу виконання компіляції чи окремих його етапів може бути використано вікно стану (**Status**) менеджеру проекту. Щоб його активувати потрібно виконати команду *View>Utility Windows>Status* (рис. 39). В полі **Progress%** динамічно відображається хід виконання окремих етапів компіляції з вказанням відсотку виконаної роботи. Це ж поле одночасно відображається у вікні задач зліва від назви виконуваного етапу компіляції, а з права від назви, в полі **Time**, вказується час виконання етапу компіляції, який також одночасно відображається в однойменному полі у вікні **Status.**

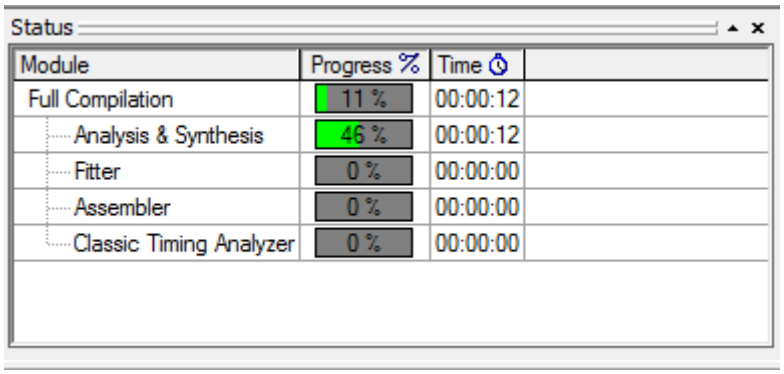


Рисунок 39. Вікно стану менеджера проекту

Після успішного завершення етапу компіляції на екран виводиться відповідне повідомлення (рис. 40), а у вікні задач виконаний етап відображається зеленим світлом та зліва від його назви встановлюється зелена галочка.

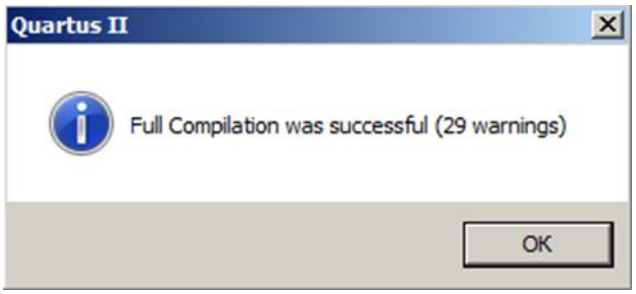


Рисунок 40. Вікно повідомлення про успішне завершення компіляції проекту

У вікні повідомлень менеджера проекту відображається інформація про хід виконання компіляції, включаючи попередження та повідомлення про помилки. Повідомлення про успішне виконання кроків компіляцію виводиться зеленим кольором і починається зі слова **Info**, попередження виводяться синім кольором і починаються зі слова **Warning**, помилки виводяться червоним кольором та починаються зі слова **Error**.

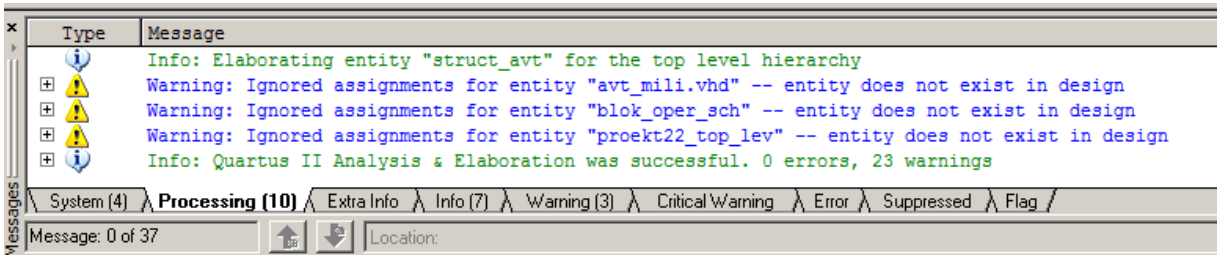


Рисунок 41. Вікно повідомлень менеджера проекту

На рисунка 41 наведено приклад вікна повідомлень менеджера проекту. У цьому вікні використовується прийом закладок, який дозволяє

шляхом клацання клавіші миші по закладці, відкрити відповідну сторінку. З допомогою закладки **Flag** можна помітити окреме повідомлення прапорцем. З допомогою закладки **Suppressed** можна приховати появу деяких повідомлень. Використання цієї можливості може знадобитись у тих випадках, коли повідомлення проаналізовано, і вивід його на екран більше не потрібен.

Закладки **Error, Critical Warning, Warning** дозволяють відкрити сторінки з повідомленнями про помилки, критичні попередження та попередження, відповідно.

Об’єктом компіляції завжди є модуль верхнього рівня. Тому для виконання компіляції модуля нижчого рівня, необхідно оголосити його модулем верхнього рівня. Зробити це можна з допомогою закладок **Hierarchy** та **Files** у вікні навігатора проекту, з використанням контекстно-залежного меню, в якому слід виконати команду **Set as Top Level Entity**.

Відображення результатів компіляції проекту

В склад пакету середовища Quartus II входять засоби, які дозволяють наглядно пред’явити звіт про результати компіляції проекту, виконати аналіз даного звіту та внести деякі зміни в результат компіляції проекту.

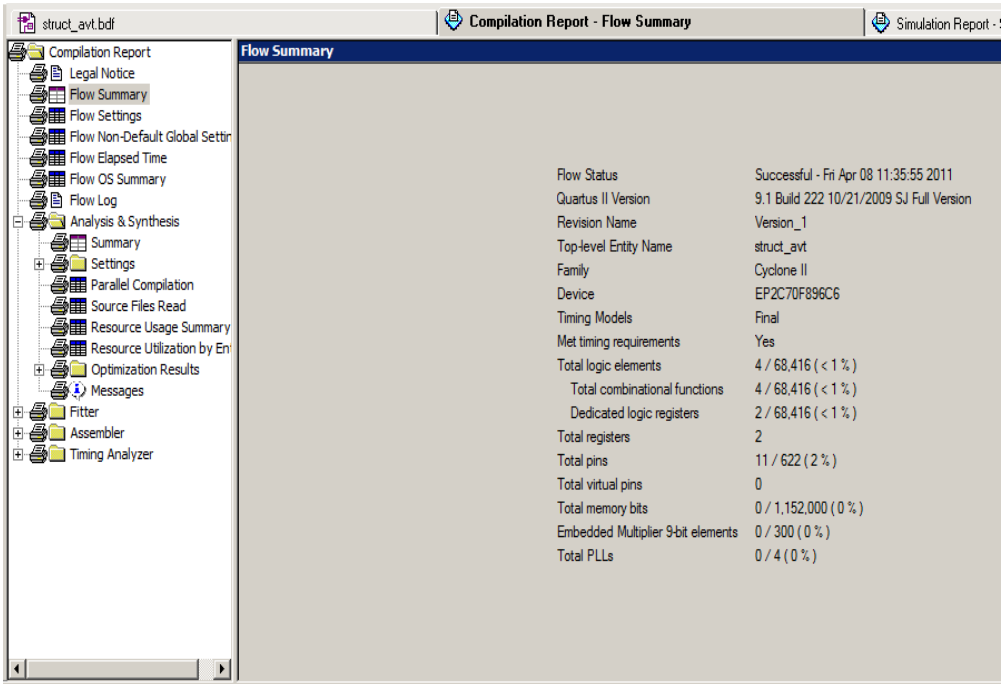


Рисунок 42. Вікно звіту про компіляцію (розділ **Flow Summary**)

Звіт про результат компіляції з’являється зразу після завершення компіляції проекту. Він розташовується в головному вікні менеджера

проекту і складається з двох частин (рис. 42). В лівій частині показана ієрархічна структура звіту, де інформація про виконання окремих етапів компіляції знаходиться у відповідних папках, а в правій частині – сам звіт. По замовчуванню, в правій частині виводиться узагальнена інформація про результати компіляції проекту. Щоб переглянути інші розділи звіту, потрібно в лівій частині, з допомогою клацання клавіші миші, вибрати необхідний розділ звіту, котрий буде відображений в правій частині вікна.

Вікно із звітом може бути згорнуте, розгорнуте чи закрите стандартним чином. Його можна відкрити або виконанням команди **Processing>Compilation Report**, або клацанням миші по піктограмі  на панелі інструментів.

Інформація про звіт також доступна в текстовому вигляді в робочій папці проекту у файлах **<project_name>.fit.rpt** чи **<project_name>.map.rpt**.

Щоб проглянути вікно з розділом звіту про використовувані ресурси кристалу для деякого проекту (рис. 43) потрібно розкрити папку **Analysis & Synthesis** в лівій частині вікна звіту з допомогою клацання миші по значку «+», поруч з ім'ям папки. При виборі **Resource Usage Summary** з'являється вікно зображене на рисунку 43.

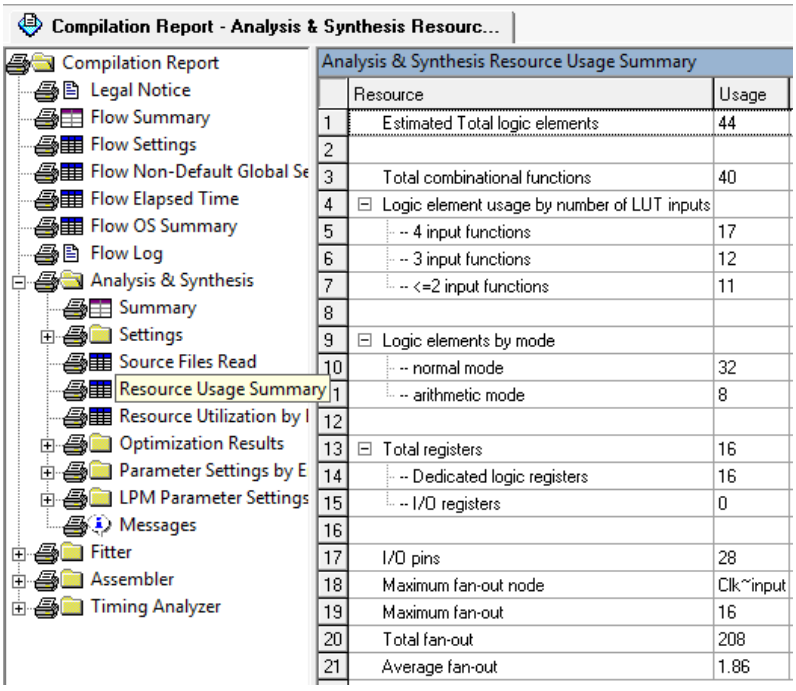
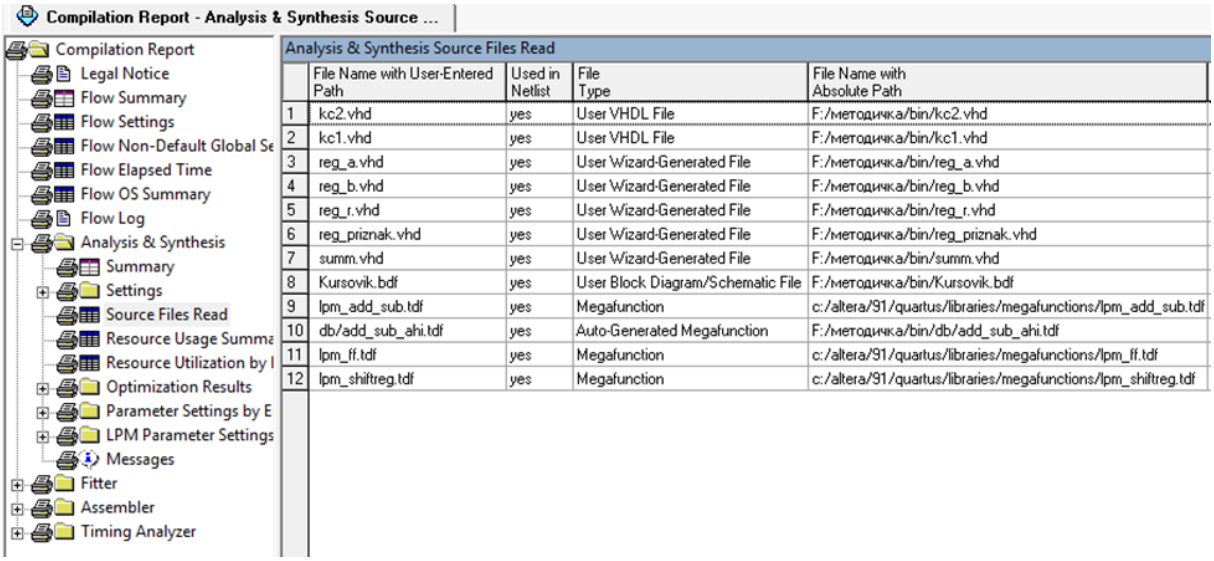


Рисунок 43. Вікно звіту про використання ресурсів кристалу

Щоб відкрити вікно, яке відображає використані в процесі компіляції вихідні файли проекту (рис. 44), необхідно клацнути мошкою по рядку **Source Files Read** в розділі звіту із папки **Analysis & Synthesis**.



	File Name with User-Entered Path	Used in Netlist	File Type	File Name with Absolute Path
1	kc2.vhd	yes	User VHDL File	F:/методичка/bin/kc2.vhd
2	kc1.vhd	yes	User VHDL File	F:/методичка/bin/kc1.vhd
3	reg_a.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_a.vhd
4	reg_b.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_b.vhd
5	reg_r.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_r.vhd
6	reg_priznak.vhd	yes	User Wizard-Generated File	F:/методичка/bin/reg_priznak.vhd
7	summ.vhd	yes	User Wizard-Generated File	F:/методичка/bin/summ.vhd
8	Kursovik.bdf	yes	User Block Diagram/Schematic File	F:/методичка/bin/Kursovik.bdf
9	lpm_add_sub.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_add_sub.tdf
10	db/add_sub_ahi.tdf	yes	Auto-Generated Megafunction	F:/методичка/bin/db/add_sub_ahi.tdf
11	lpm_ff.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_ff.tdf
12	lpm_shiftreg.tdf	yes	Megafunction	c:/altera/91/quartus/libraries/megafunctions/lpm_shiftreg.tdf

Рисунок 44. Вікно звіту з вихідними файлами проекту.

В цьому вікні звіту приводиться список всіх логічних файлів, як користувальницьких, так і бібліотечних, які використовуються в процесі компіляції проекту. Також вказуються типи файлів та папки, в яких вони зберігаються.

Для перегляду внутрішнього подання проекту використовується засіб Quartus II, зване **Netlist Viewers**. За допомогою цього засобу можна переглядати реалізацію проекту на програмованому кристалі на різних рівнях абстракції. Так, **RTL Viewer** дозволяє переглядати результати компіляції проекту на рівні регістрових передач. **Technology Map Viewer** дозволяє переглядати технологічну карту проекту, тобто його реалізацію з використанням таких ресурсів кристала, як функціональні перетворювачі (**LUT**), так і тригери. Для перегляду результатів компіляції модулів проекту, описаних у вигляді кінцевого автомата, призначене засіб ***State Machine Viewer***.

Для перегляду та аналізу результатів компіляції проекту призначене засіб **Chip Planner**, для виконання аналізу та внесення змін в результати компіляції - засіб **Resource Property Editor**.

Хід проектування

1. Компілювання проекту

1.1. Для компіляції проекту потрібно обрати команду **Start Compilation** в меню **Processing** або натиснути кнопку  , розташовану на

робочій панелі. Після завершення процесу компіляції відкриється діалогове вікно. Натисніть **ОК**.

2. Перегляд інформації у вікні звіту компілятора (Compilation Report)

Звіт компілятора надає повну інформацію про результат опрацювання проекту. З його допомогою можна визначити, як компілятор опрацював проект та які отримано результати. Кожна окрема утиліта, яка входить в склад компілятора, оновлює інформацію в своїй директорії.

Вікно **Compilation Repor**t відкривається, коли запущено один із додатків, які входять до складу компілятора. По закінченню роботи компілятора відкривається вікно **Flow Summary** (рис. 45).

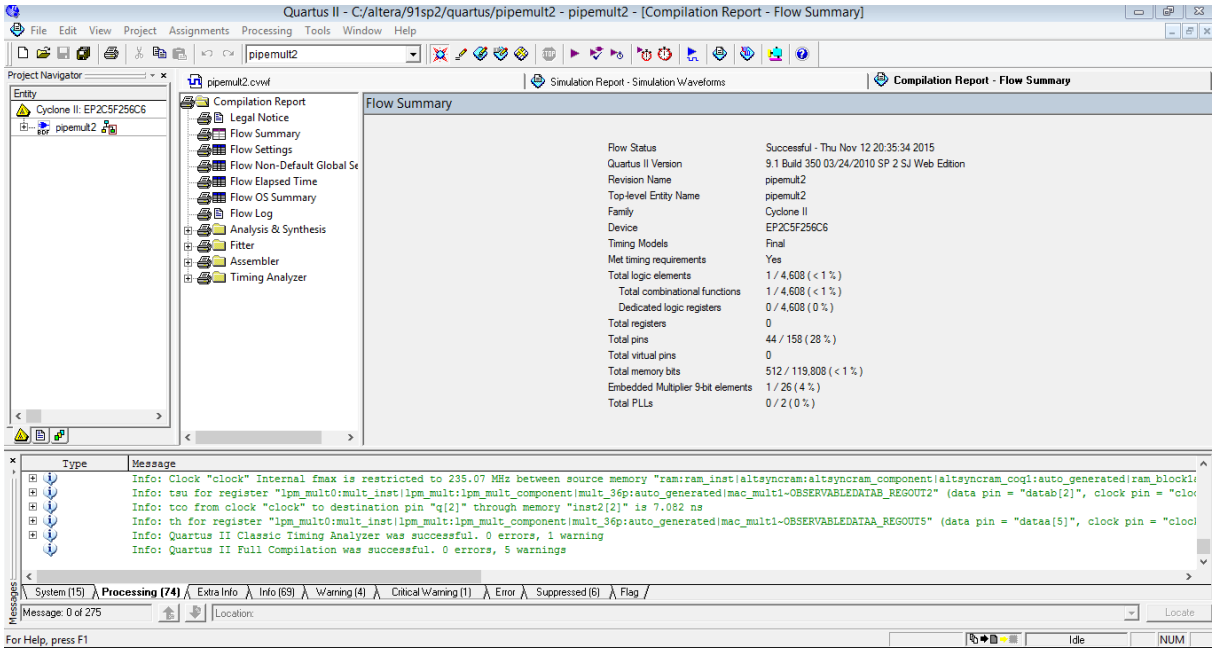
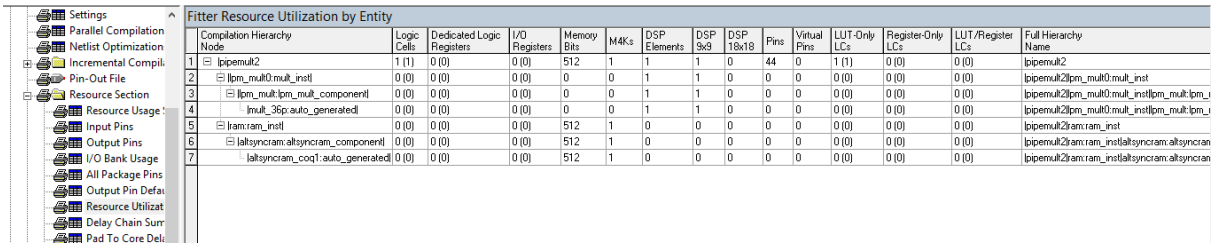


Рисунок 45. Вікно звіту компіляції

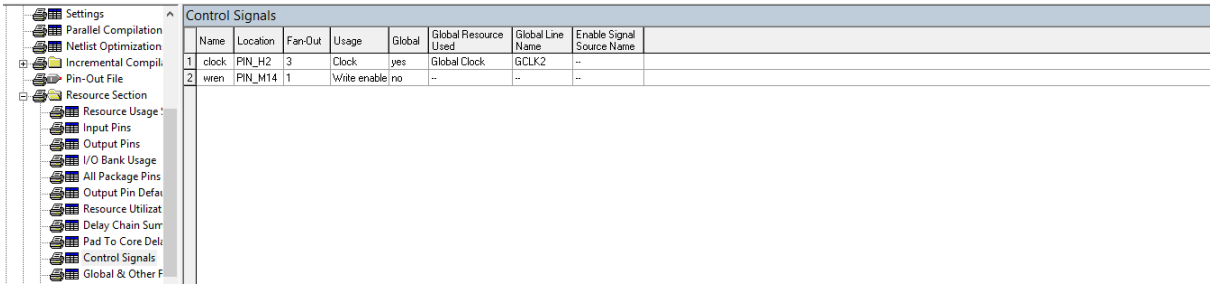
2.1. Із розділу **Flow Summary** вікна **Compilation Report**, потрібно виписати отримані значення **Total logic elements, total memory bits, embedded multiplier 9-bit elements** та **total pins** (сформувати таблицю та внести у звіт).

Аналіз результатів показує, що для реалізації проекту були задіяні спеціалізовані ресурси (тобто, вбудована пам'ять, вбудований помножувач) та додаткова логіка.

2.2. Розкрийте папку **Fitter** у вікні **Compilation Report**. Оберіть розділ **Resource Section**. Із таблиці **Resource Utilization by Entity**, випишіть дані про використовувані ресурси для реалізації блоків **mult та ram**.



2.3. Із розділу **Control Signals** випишіть основні керуючі сигнали та їх коефіцієнти розгалуження.

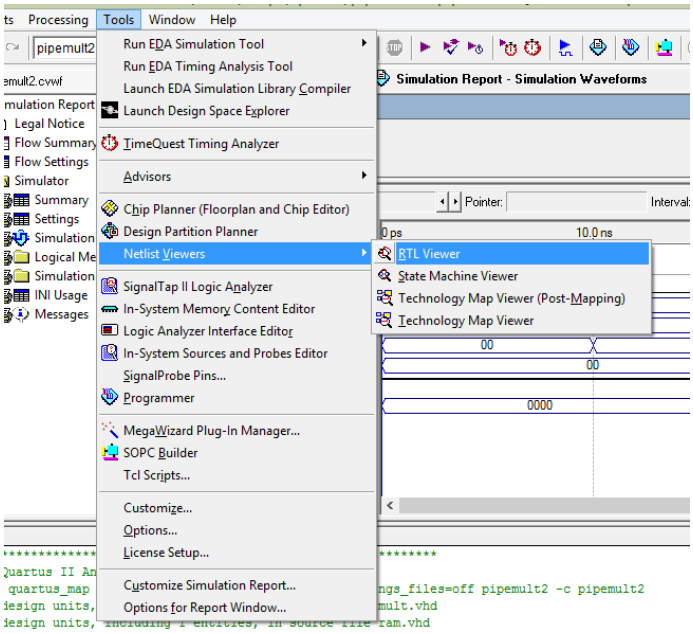


Хоч сигнал в даному проекті поступає більш ніж на 3 регістри, коефіцієнт розгалуження, рівний 3, вказує на кількість задіяних архітектурних блоків: 1 блок пам'яті, 1 блок помножувача, 1 логічна комірка.

3. Аналіз логічної реалізації проекту з допомогою RTL Viewer

Утиліта **RTL Viewer** дозволяє представити логічну реалізацію проекту в графічному вигляді. Це дуже корисний інструмент для аналізу результатів синтезу **HDL** проектів.

3.1. В меню **Tools** оберіть утиліту **RTL Viewer** (в списку **Netlist Viewers**).



На рисунку 46 зображено графічне відображення логічної реалізації проекту. Тут присутні контакти вводу/виводу, блоки **mult** та **ram**, а також вихідні регістри. Слід звернути увагу на те, що вхідний регістр є зовнішнім елементом по відношенню до блоку пам’яті.

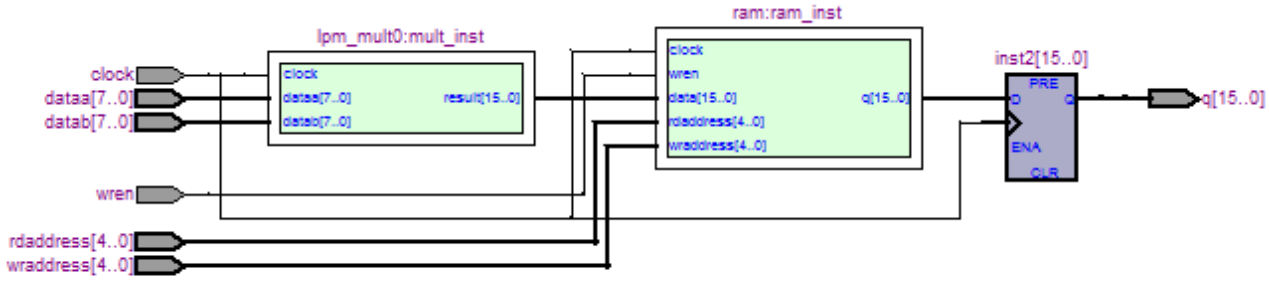
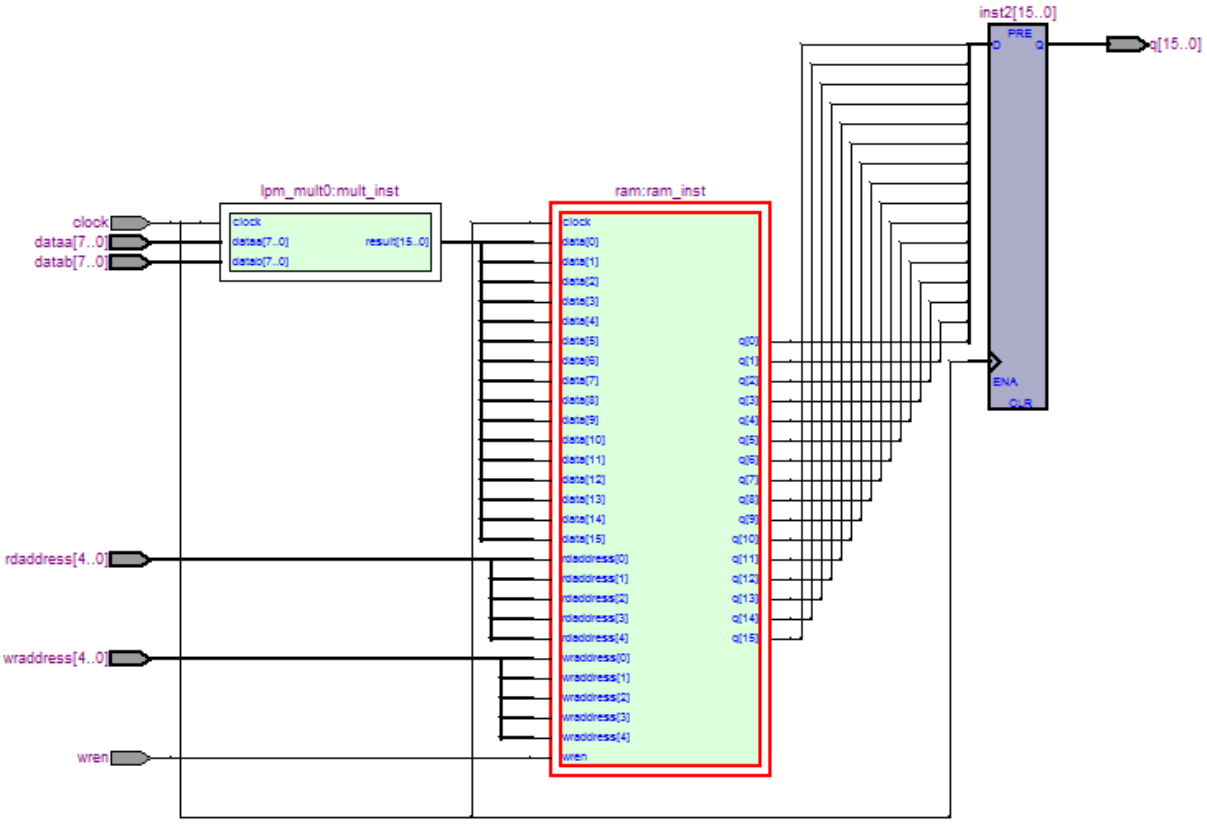


Рисунок 46. Графічне відображення логічної реалізації проекту

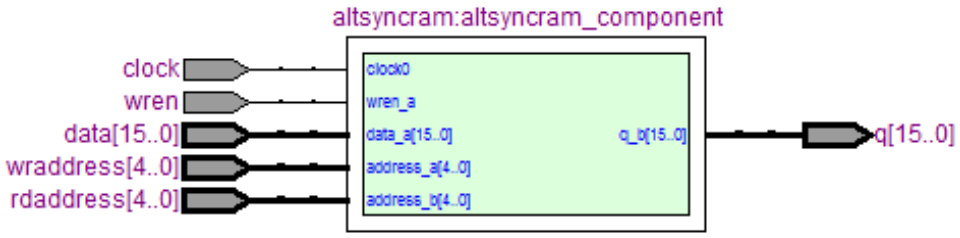
3.2. Виділіть блок **ram,** натисніть праву клавішу миші і в контекстному меню виберіть команду **Ungroup Selected Nodes**.



Тепер блок пам’яті відображається із всіма входами та виходами. Це корисно, коли потрібно розглянути, яким чином з’єднуються окремі сигнали. Ця операція доступна для вхідних блоків та контактів вводу/виводу.

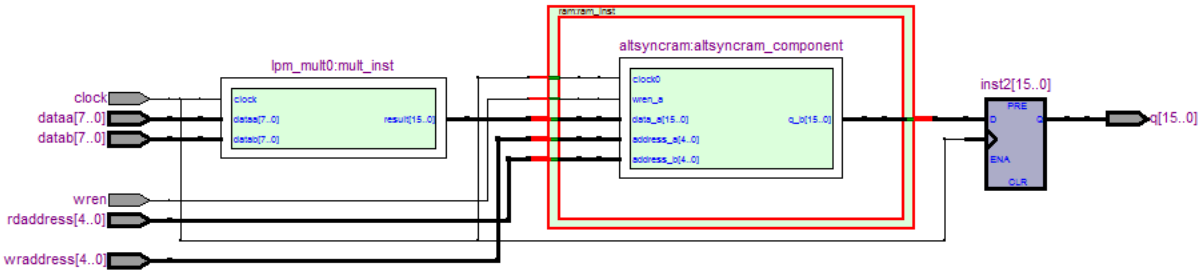
3.3. Повторно виділіть блок **ram,** натисніть праву клавішу миші і в контекстному меню виберіть команду **Group Related Nodes**.

3.4. Двічі клацнувши клавішою миші на блок **ram**, здійснюється перехід на нижчий рівень ієрархії – до блоку **ram**. Звідси видно, що для реалізації блоку **ram** використовується одна мегафункція **altsyncram**.



Щоб повернутись до верхнього рівня ієрархії потрібно двічі клацнути на пустому місці графічного вікна.

3.5. Повторно виділіть блок **ram**. Натисніть праву клавішу миші та оберіть команду **Display Content**.



Ця команда дозволяє відобразити нижній рівень ієрархії в рамках даного рівня. Зелений блок **altsyncram**, виділений червоним контуром, показує границю ієрархічних рівнів. Повторення даного алгоритму приведе до переходу до найнижчого рівня ієрархії для блоку **altsyncram_coq1**, та дає можливість побачити, що він складається із 16 функціональних однобітних елементів RAM. Все це функціональне представлення проекту, а не його практична реалізація в мікросхемі **Cyclone II**.

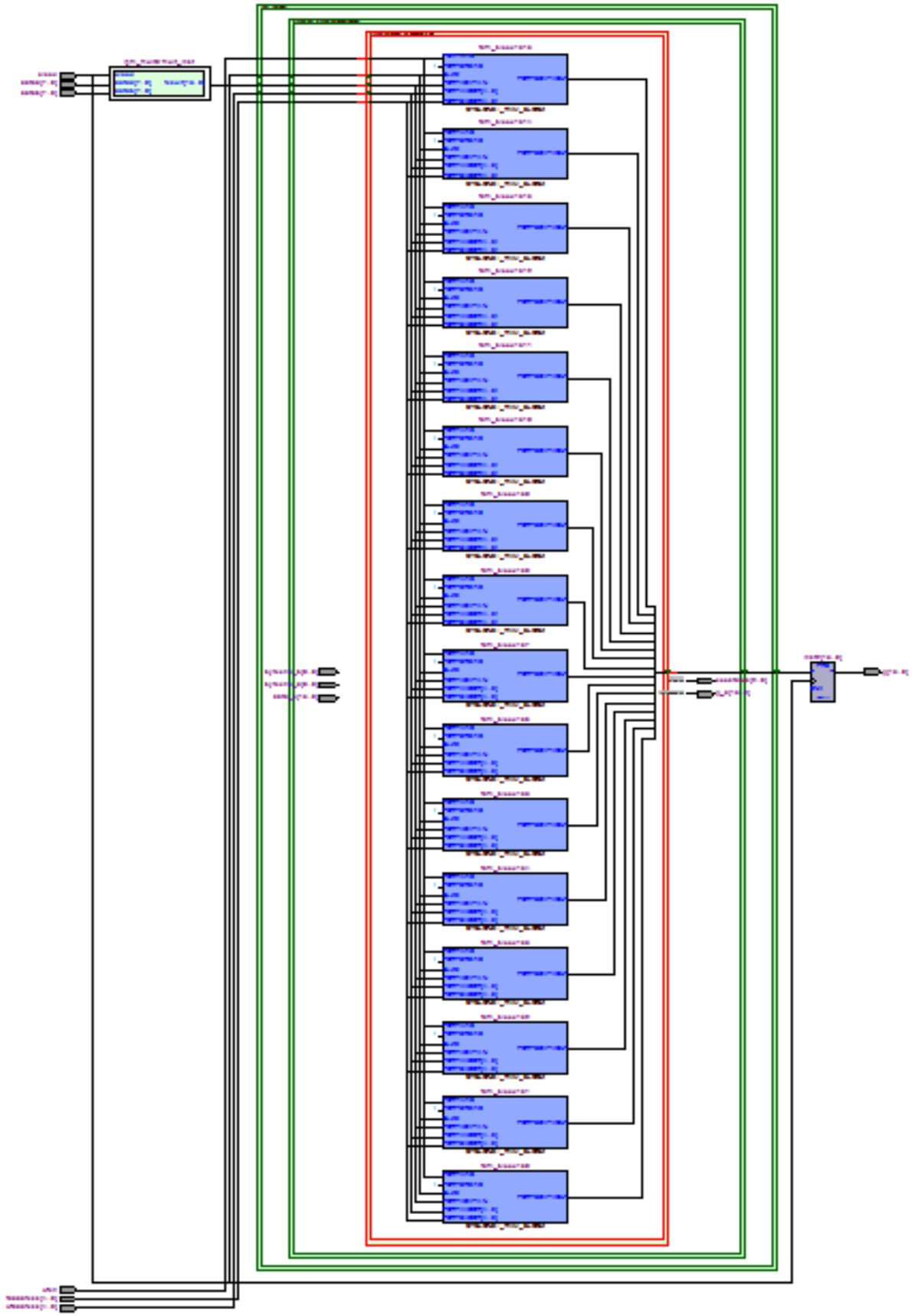


Рисунок 47. Функціональне представлення проекту у середовищі Quartus II

Слід звернути увагу на те, що вікно ієрархії проекту **Hierarchy List** утиліти **RTL Viewer** також відображає ієрархічну структуру блоку **ram**. Для кожного рівня ієрархії додатково відображаються імена контактів та внутрішніх ланцюгів (рис. 48).

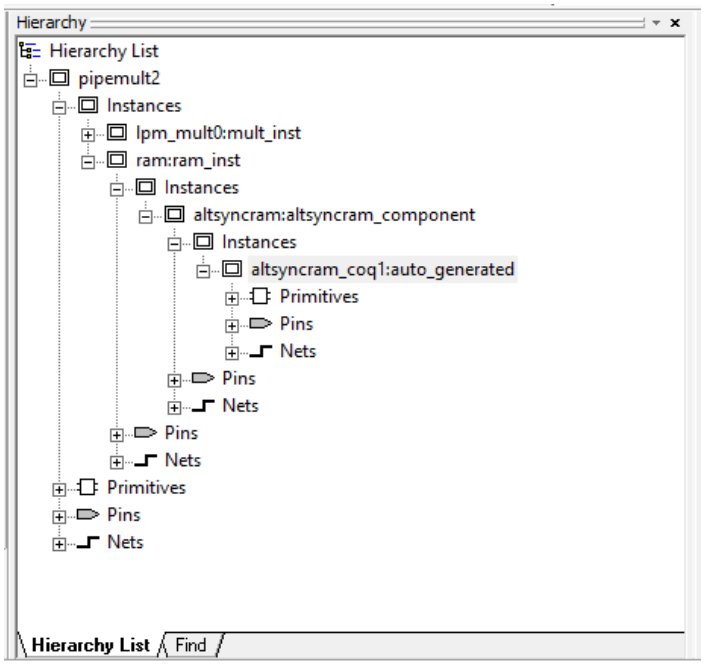
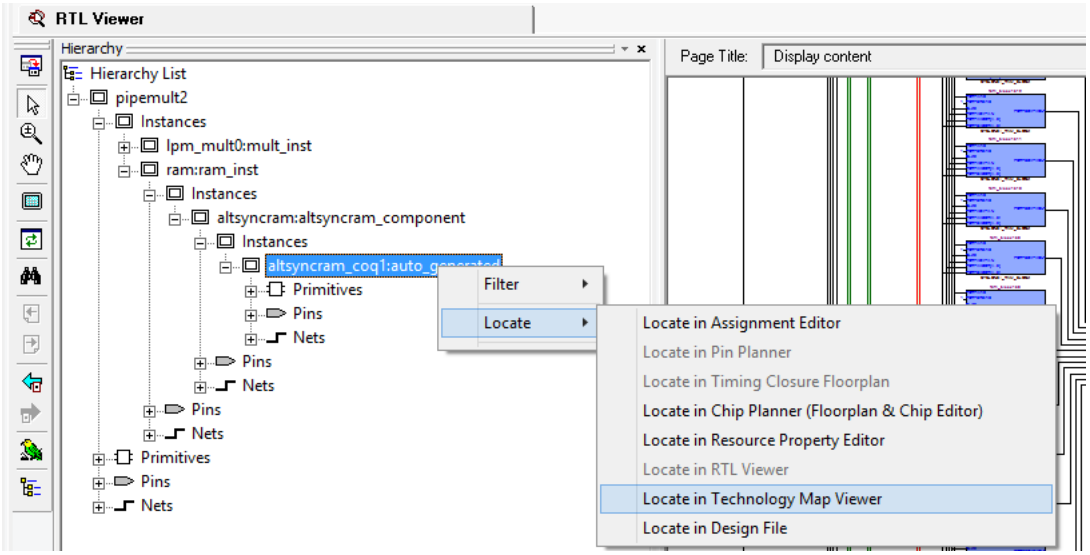


Рисунок 48. Відображення імен контактів та внутрішніх ланцюгів для кожного рівня ієрархії

4. Перевірка фізичної реалізації блоку ram з допомогою Technology Map Viewer.

Утиліта **Technology Map Viewer** дозволяє переглядати фактичну реалізацію проекту та використовувані ресурси FPGA/CPLD. Ця утиліта використовується в якості допоміжного інструменту в процесі відлагодження проекту для аналізу змін у використовуваних ресурсах (зміни налаштувань проекту).

4.1. У вікні **RTL Vie**wer необхідно перейти на ієрархічний рівень, який відображає блок **ram** (модуль altsyncram_coq1). Виділіть його. Натисніть праву клавішу миші та оберіть в контекстному меню команду **Locate -> Locate in Technology Map Viewer**



Відкриється вікно **Technology Map Viewer** із зображенням модуля **altsyncram_coq1**.

4.2. Перейшовши на нижній рівень ієрархії для модуля **altsyncram_coq1**, використовуючи будь який із запропонованих раніше способів, отримаєм наступне зображення (рис. 49)

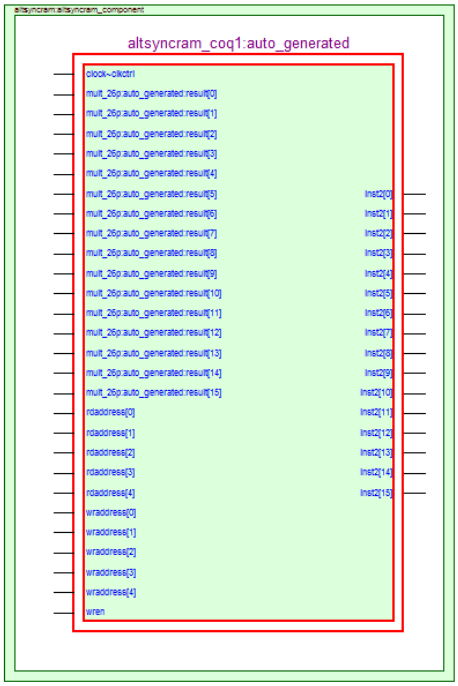


Рисунок 49. Нижній рівень ієрархії модуля altsyncram_coq1

На відміну від відображених в **RTL Viewer** 16 однобітних функціональних блоків пам’яті, **Technology Map Viewer** відображає

фактично використовуваний ресурс мікросхеми – єдиний блок вбудованої пам’яті (**ram_block1a0**).

Натисніть двічі ліву кнопку миші на блоці **ram_block1a0** для докладного аналізу способу реалізації блоку пам'яті.

Як видно, всі вхідні сигнали блоку **RAM** надходять на вбудовані регістри так само, як і всі вихідні сигнали. Попри те, що вихідні регістри відображалися за межами блоку пам’яті у вікні **RTL Viewer**, трасувальник перемістив ці регістри в блок пам’яті, щоб поліпшити його продуктивність і знизити кількість використовуваних логічних ресурсів у проекті. Такий спосіб оптимізації називається упаковкою регістрів, і трасувальник використовує його для зменшення логічних ресурсів мікросхеми. В даному випадку трасувальник видає повідомлення «**Extra Info**» (додаткова інформація) у вікні повідомлень **Message** (воно розташоване в таблиці **Suppressed**), яке вказує на те, що використовується така оптимізація.

5. Перевірте зв'язок блоку ram за допомогою редактора топології кристала Chip Planner

Редактор **Chip Planner** надає можливість перегляду фізичного розміщення логічних блоків вашого проекту на кристалі. Це дуже корисно для розуміння особливостей реалізації проекту, а також архітектурних особливостей ПЛІС FPGA і CPLD. Хоча редактор **Chip Planner** застосовується для ручного призначення логічних ресурсів, його можна використовувати і для оцінки отриманих результатів.

5.1. У вікні **Technology Map Viewer**, виділіть блок RAM (**ram block 1a0**). Натисніть праву клавішу миші і виберіть команду **Locate -> Locate in Chip Planner (Floorplan & Chip Editor)** в контекстному меню.

5.2. У вікні редактора топології кристала **Chip Planner** виберіть кнопку  (в панелі інструментів) і за допомогою правої клавіші миші зменшите масштаб зображення в кілька разів (рис. 50).

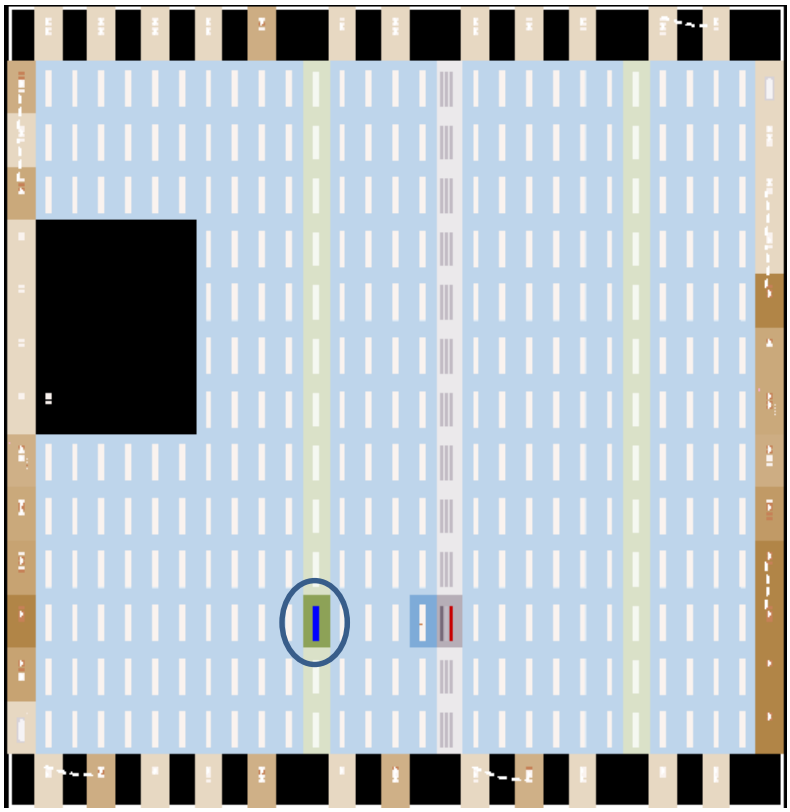


Рисунок 50. Розташування блоку ram

У вікні редактора зображено блок **ram** (виділений синім кольором) і його розміщення в мікросхемі **Cyclone II**. Решта виділені області у вікні **Chip Planner** – це ресурси, які використовуються для розміщення інших логічних блоків проекту. Так як не було зроблено ніяких призначень щодо розміщення логічних блоків, трасувальник самостійно визначав необхідні ресурси.

5.3. Якщо блок пам'яті виділений у вікні **Chip Planner**, натисніть кнопку  (**Generate Fan-In Connections**) на панелі інструментів (відображення вхідних зв'язків для виділеного блоку).

У вікні **Chip Planner** (рис. 51) відображаються ті логічні ресурси мікросхеми, від яких надходять вхідні сигнали в блок **ram** (блоки введення/виведення і помножувач) і затримки поширення сигналів по цих зв’язках.

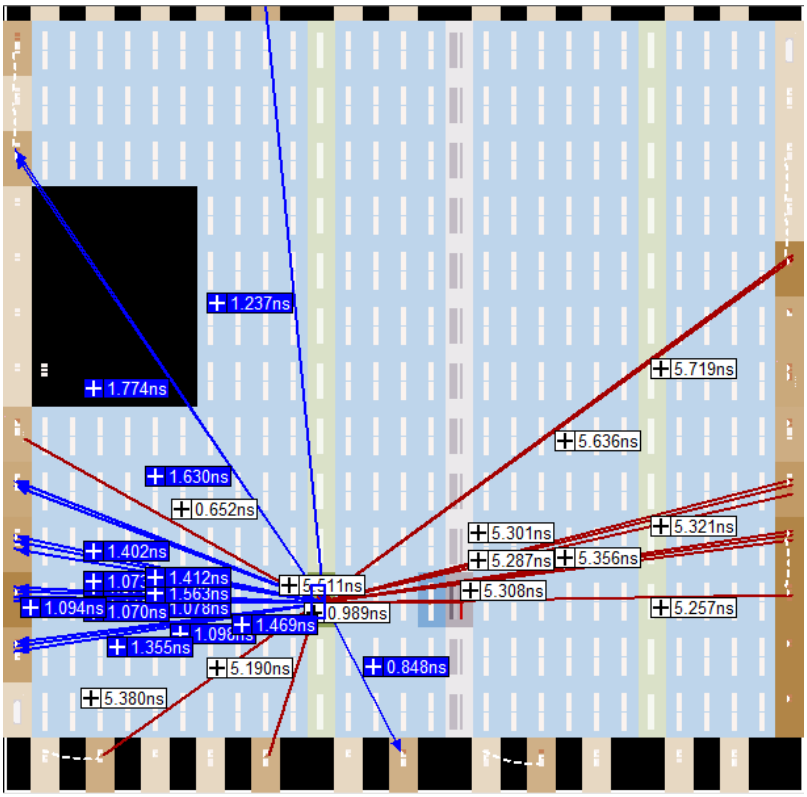


Рис. 51. Відображення логічних ресурсів мікросхеми,
до яких надходять вихідні (синього кольору) та від яких надходять вхідні (червоного кольору) сигнали в блок ram

5.4. Виділіть знову блок **ram** в **Chip Planner** (можливо доведеться зменшити масштаб зображення), і натисніть кнопку (**Generate Fan-Out Connections**) на панелі інструментів (рис. 51, зв'язки синього кольору).

5.5. Скасуйте команди **fan-in/fan-out**, натиснувши ліву кнопку миші на будь-якому невиділеному блоці, а потім натисніть (**Clear Unselected Connections/Paths**) на панелі інструментів.

5.6. Закрийте вікна **RTL Viewer**, **Technology Map Viewer** і **Chip Planner**.

Завдання на лабораторну роботу 7

5. Ознайомитись із теоретичними відомостями.
6.
7.
8. Зробити висновки. Результати у вигляді скріншотів представити у звіті.

Контрольні питання

6.
7.
8.
9.
10.

Лабораторна робота № 9

Призначення контактів вводу/виводу в проекті

Мета роботи: Освоїти використання утиліт **Pin Planner** і **Back-annotation** для призначення контактів

Теоретичні відомості

Функціонування пристрою на ПЛІС багато в чому визначається установками компілятора та призначеннями, які виконуються для визначення виводів мікросхеми та стилів синтезу проекту. Етап визначення опцій синтезу проходить за алгоритмом, наведеним на рисунку 52. Установки, виконані за допомогою діалогу **Settings**, редактора призначень (**Assignment Editor**), планувальника виводів (**Pin Planner**) і часових аналізаторів підсумовуються у файлі установок пакету Quartus II (Quartus II Settings File – *.qsf). Ці дані, нарівні з файлами, що описують проект, визначають результат компіляції.

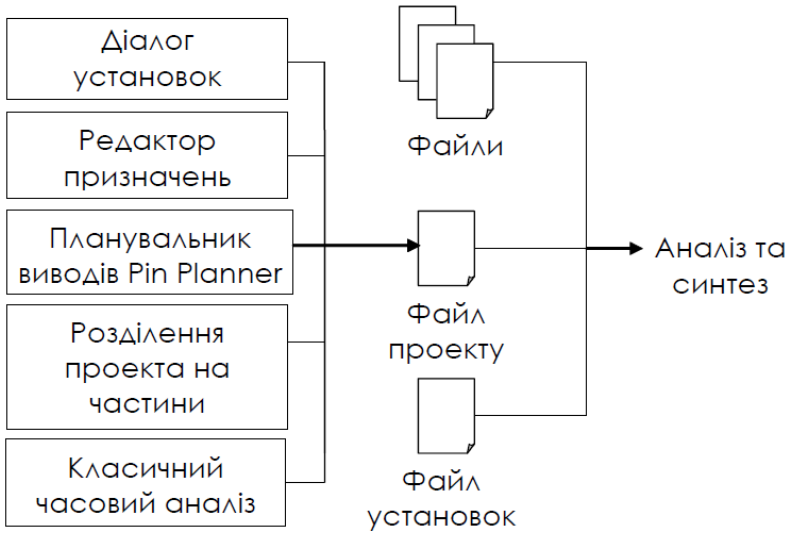


Рисунок 52. Визначення параметрів проекту

Призначення виводів у середовищі **Quartus II** може бути виконано декількома способами. Найбільш часто використовуються наступні методи:

- з використанням планувальника виводів (**Pin Planner**);
- імпорт із таблиці в **CSV** форматі;
- редагуванням файлу призначень проекту (**QSF**);
- за допомогою програми.

Планувальник виводів – це інтерактивний засіб, який може бути відкритий за допомогою вибору пункту меню **Assignments → Pin Planner** або кнопки (рис. 53). Його вікно складається із трьох частин:

- списку всіх виводів;
- списку груп виводів;
- зображення мікросхеми – виду згори (**Top view**) або виду знизу (**Bottom view**).

Список виводів містить наступні поля:

- **Node Name** – ім’я виводу.
- **Direction** – напрямок передачі даних: вхід, вихід або двонаправлений.
- **Location** – розташування, тобто номер виводу в корпусі.
- **I/O Bank** – банк вводу–виводу.
- **Vref Group** – опорна напруга для групи виводів.
- **I/O Standart** – стандарт вводу–виводу.
- **Reserved** – резервування, тобто вказівка на те що необхідно робити з тими виводами, які не використовуються.
- **Group** – група виводів.

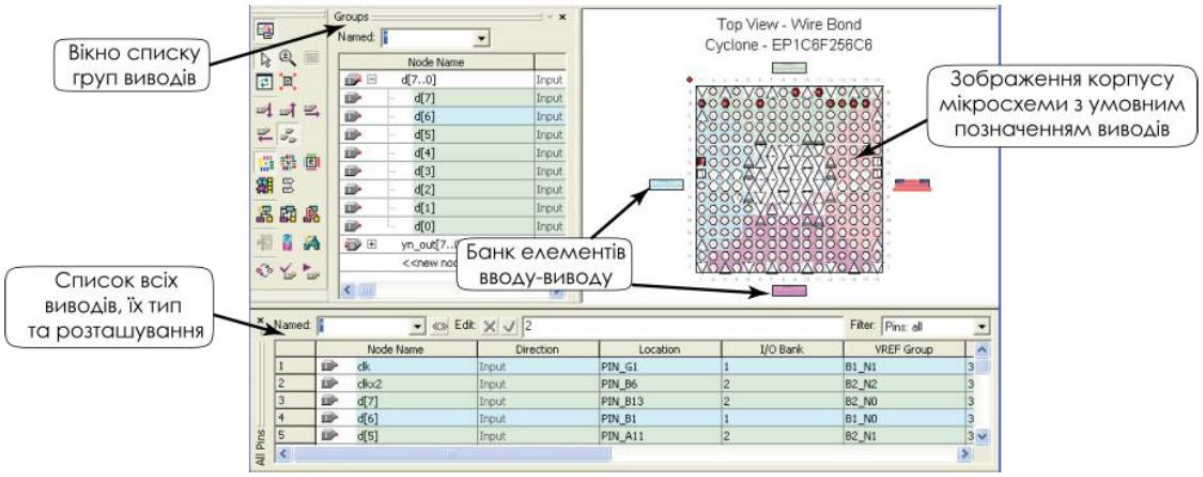


Рисунок 53. Вікно редагування виводів проекту

Порядок призначення виводів у середовищі **Quartus II** за допомогою **Pin Planner**.

1. Відкрити проект.
2. Підготувати планувальник до роботи.

2.1. Відкриваємо планувальник виводів за допомогою меню **Assignments → Pin Planner**.

2.2. Для вигляду мікросхеми встановлюється вид згори (**Top view**) за допомогою пункту меню **View → Show → Package Top**.

2.3. Обирається режим відображення банків вводу–виводу (**Show I/O bank**) – кнопка  натиснута.

3. Призначення виводів.

3.1. У вікні списку груп виводів (**Group**) планувальника **Pin Planner** обирати групу входів, наприклад, data[7..0]. Утримуючи цю групу, за допомогою лівої клавіші перетягуємо її на позначення банку елементів вводу–виводу **I/OBank_2** у вікні зображення мікросхеми. При цьому **Quartus II** автоматично призначить виводам проекту доступні виводи мікросхеми. Такий метод використовується в тому випадку, коли ще немає топології друкованої плати, тобто не існує жорсткої прив'язки виводів проекту.

3.2. У тому випадку, коли потрібно призначати окремі виводи, можна користуватися наступними способами:

3.2.1. Вибрати необхідний вивід у списку всіх виводів і перетягнути на потрібний вивід у вікні зображення мікросхеми.

3.2.2. Вибрати необхідний вивід у списку всіх виводів та у таблиці параметрів зробити подвійне клацання по полю **Location** (рисунок 54). В списку вибрати потрібний вивід.

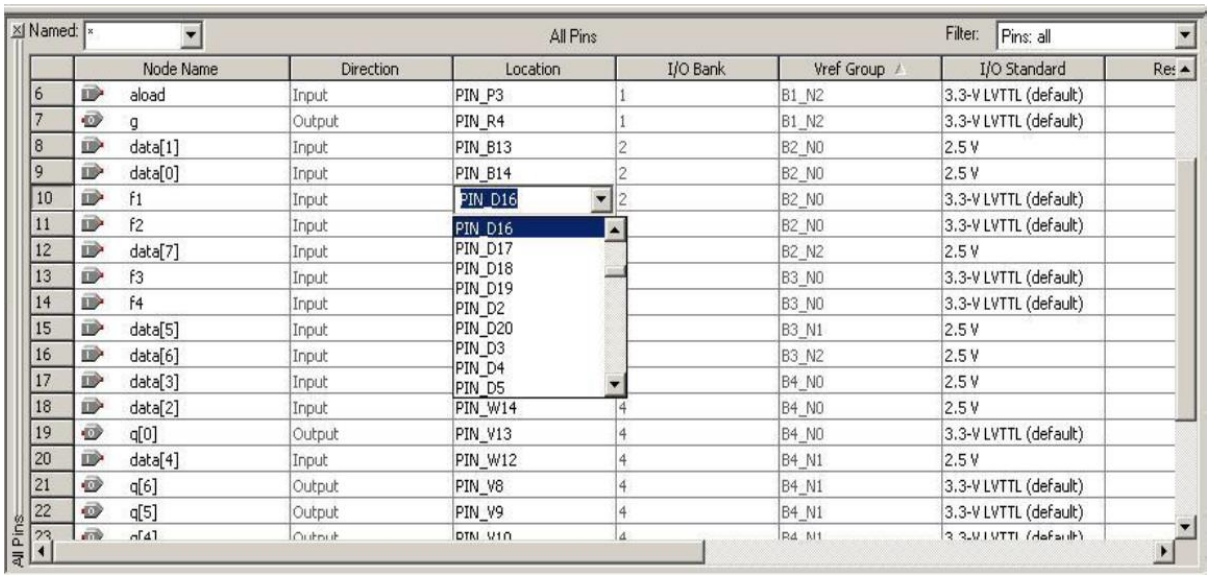


Рисунок 54. Таблиця виводів мікросхеми

4. Призначення властивостей виводів. Визначення властивостей перш за все передбачає визначення стандарту вводу–виводу для одного або

декількох виводів. Для доступу до вікна властивостей необхідно виконати одну з описаних дій:

– У вікні списку груп виводів (**Group**) натиснути праву кнопку на шині і вибрати пункт **Node Properties**, що призведе до відкриття вікна, показаного на рисунку 55.

– У вікні списку всіх виводів вибрати один або кілька виводів та у контекстному меню вибрати пункт **Node Properties**.

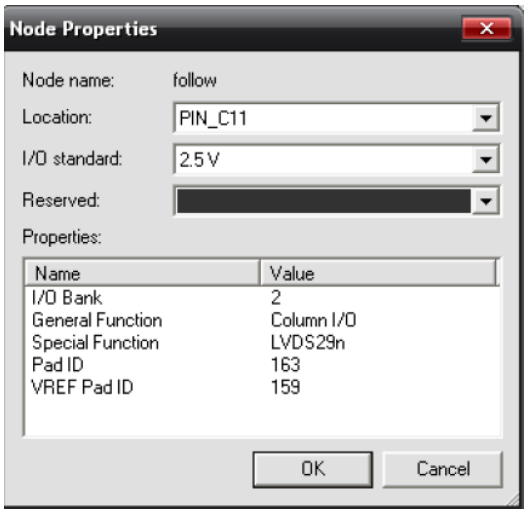


Рисунок 55. Вікно властивостей виводу

Вікно властивостей містить наступні поля:

- **Location** – вказується розташування виводу. Для декількох виводів може вказуватися банк елементів введення – виведення.
- **I/O standart** – стандарт введення – виведення.
- **Reserved** – зарезервований. Використовується в тому випадку, коли необхідно даний вивід зарезервувати на майбутнє.
- **Properties** – список властивостей виведення.

5. У відкритому вікні необхідно вибрати потрібний стандарт введення-виведення. Для мікросхем сімейства Cyclone стандарт вводу-виводу – 3,3 V LVTTL.

Перевірка правильності призначень виводів робиться або автоматично при повній компіляції проекту, або запуском модуля компілятора, відповідального за таку перевірку.

Для перевірки призначення виводів необхідно вибрати пункт меню **Processing → Start → Start I/O Assignment Analysis** або натиснути кнопку  на панелі інструментів планувальника виводів. Це дозволить перевірити правильність призначень без повної компіляції. Для запуску перевірки

призначення виводів необхідно тільки зробити призначення виводів (напрямок передачі даних, стандарти введення – виведення, банки).

Результати перевірки можуть бути переглянуті в розділі **Fitter** звіту компіляції. Для цього можна скористатися пунктами **Pin–Out File**, **Resource Section**. Також необхідно ще раз перевірити стан невикористаних виводів, що можна подивитись у закладці **Unused Pins** діалогу **Device and Pin Options** з закладки **Assignments → Device**.

Хід проектування

1. За допомогою утиліти Pin Planner виконайте призначення контактів і стандарту введення / виводу.

1.1. У меню **Assignments** виберіть команду **Pin Planner** (рис. 56).

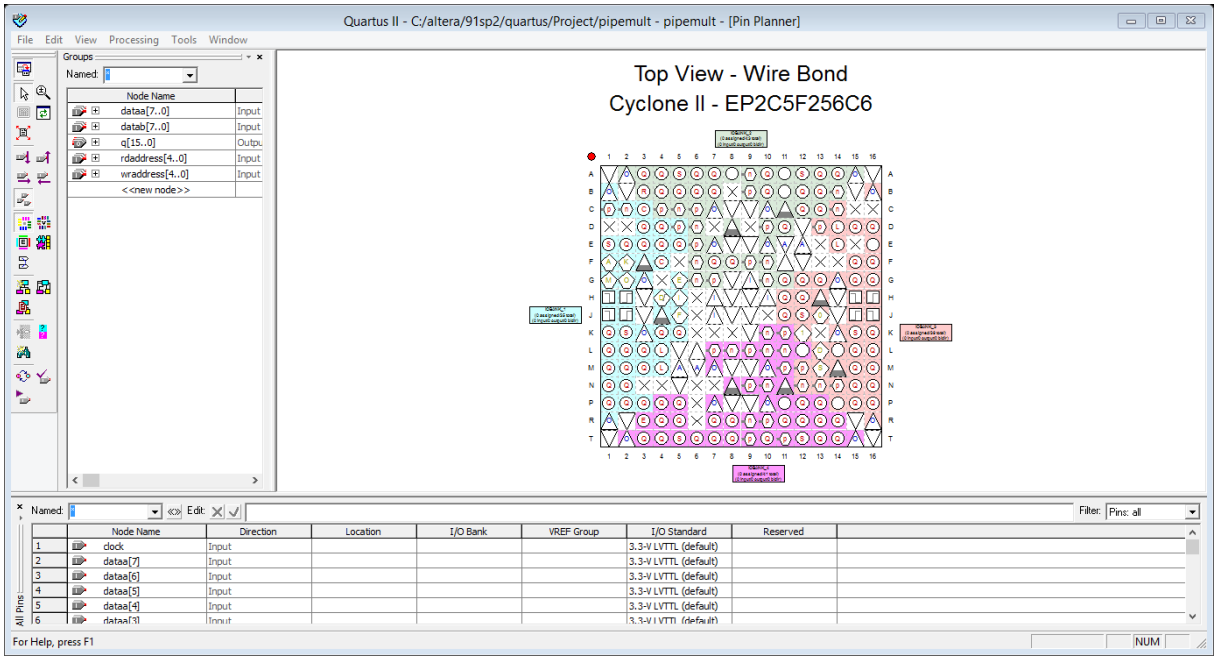


Рисунок 56. Вигляд основного вікна утиліти **Pin Planner**

1.2. У вікні мікросхема відображається в режимі **Top View**. Якщо це не так, то слід оберати команду **Package Top** в меню **View**.

1.3. У панелі інструментів утиліти **Pin Planner** слід обрати режим **Show I/O Banks** (кнопка активна). Якщо це не так, то необхідно вибрати її. Цей же режим можна задати з меню **View**.

1.4. Виберіть режим інтерактивної перевірки призначень контактів введення\виведення **Live I/O**. Для цього слід натиснути кнопку на панелі інструментів або виберіть команду **Enable Live I/O Check** з меню **Processing** у вікні **Pin Planner**.

1.5. У меню **View** (вікно **Pin Planner**), виберіть команду **Live I/O Check Status Window**.

Ця команда відкриває невелике вікно, в якому відображається інформація про помилки та попередження при призначенні контактів введення/виведення. Звертайте на нього увагу в процесі виконання завдання. Якщо з'являється вказівка на помилку або видається попередження, перевірте вікно **Messages** в основному вікні **Quartus II**. Попередження (**unset junction operating temperature** і **Reserve All Unused Pins setting**) можуть бути проігноровані.

1.6. У вікні **Groups List** (**Pin Planner**), виділіть вхідну шину **dataa**. Потім, утримуючи ліву кнопку мишки, перенесіть шину **data**a з вікна **Groups List** в секцію **IOBANK_2** (рис. 57).

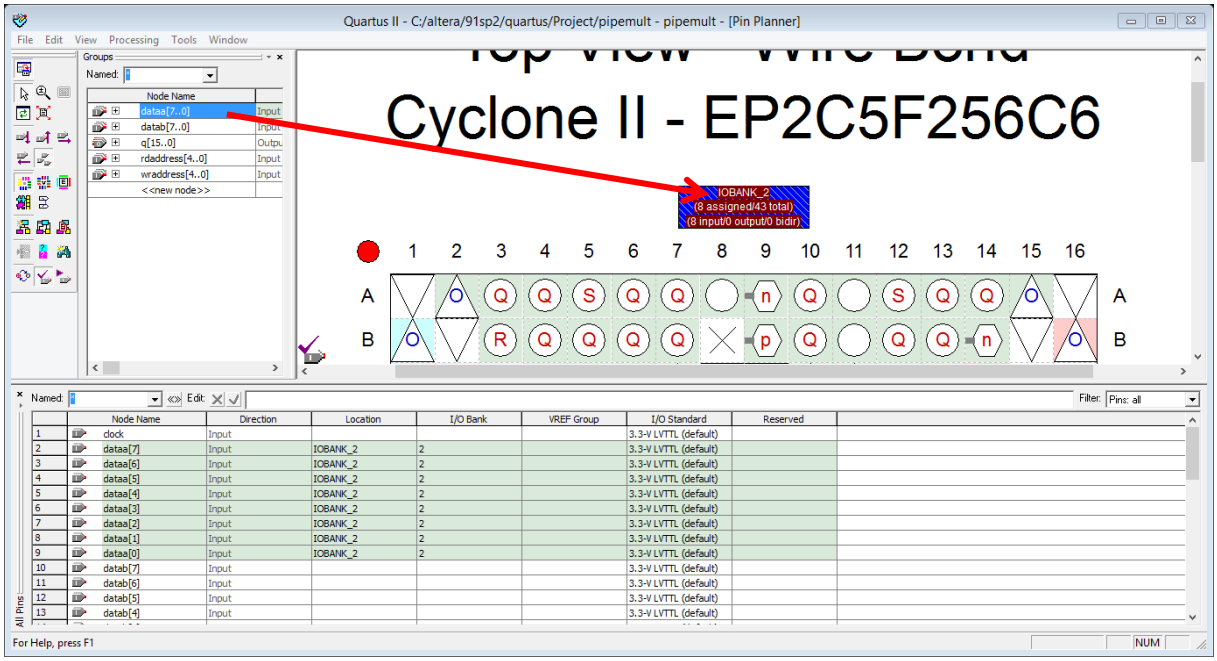


Рисунок 57. Вікно утиліти **Pin Planner**

Зображення секції **IOBANK_2** має змінитися, і буде вказувати на те, що банк **I/O Bank 2** містить 8 призначених контактів з 43. Призначення сигналів банку введення/виведення дозволяє трасувальникові розмістити дані сигнали в будь-якому місці в межах зазначеного банку.

1.7. У вікні **Groups List**, натисніть праву клавішу миші на шині **dataa** і виберіть команду **Node Properties** з контекстного меню (рис. 58). У діалоговому вікні **Node Properties** в рядку **I/O standard** для шини **dataa** встановіть значення **2.5 V**. Натисніть **OK**.

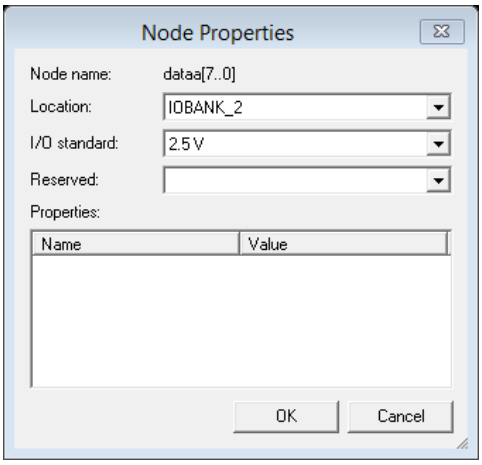


Рисунок 58. Вікно редагування властивостей виводу

Зверніть увагу, що у вікні повного переліку контактів відображаються внесені зміни стандарту введення/виведення для кожного сигналу шини **dataa** (рис. 59). Призначення контактів і стандарту введення/виведення можна виконувати і в даному вікні (**All Pins list**). Слід зауважити, що окремі призначення, зроблені у вікні **All Pins list** мають пріоритет перед груповими призначеннями, зробленими у вікні **Groups list**. Кращий спосіб виконання призначень для шин або груп сигналів – це задати призначення для назви групи (наприклад **data [7..0]**) у вікні **Groups list**.

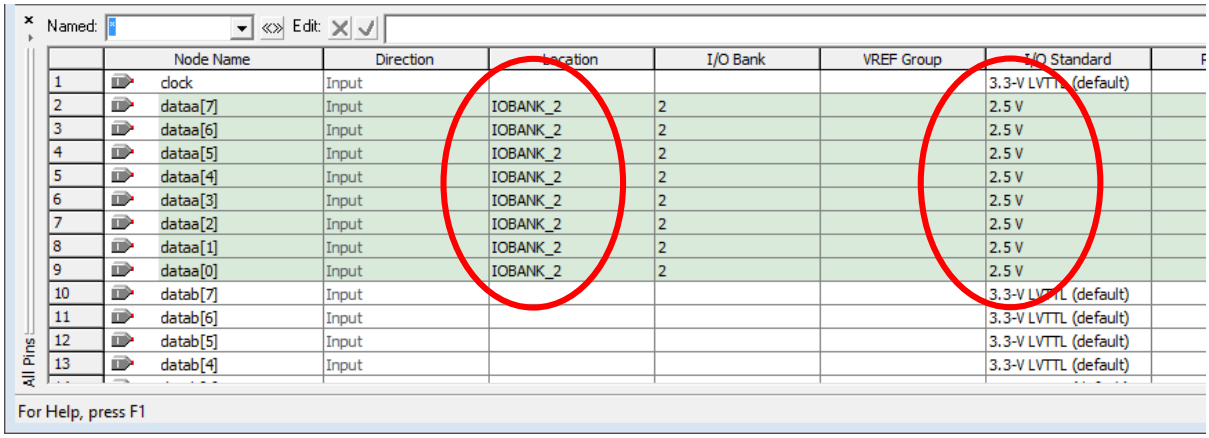


Рисунок 59. Приклад призначення контактів у вікні **All Pins list**

1.8. Використовуючи вікно **Package View** або вікно **All Pins List**, призначте для шини **datab** банк **IOBANK_2**.

1.9. Встановіть значення **I/O Standard** для шини **datab** рівне **1.8 V** за допомогою діалогового вікна **Node Properties**.

Після цього з’явиться повідомлень про помилку у вікні утиліти **Live I/O Check Status Window** і вікні повідомлень САПР Quartus II

(рис. 60). Не намагайтеся поки усунути проблему. Завдання – перевірити реакцію компілятора.

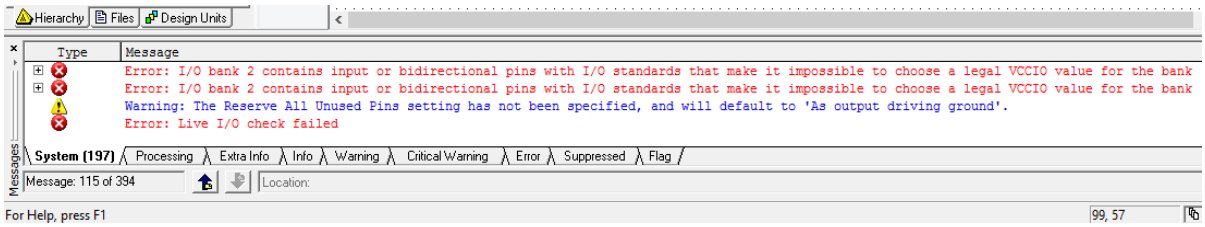


Рисунок 60. Повідомлення про помилку у вікні утиліти **Live I/O Check Status Window**

1.10. Призначте для вихідної шини **q** банк **IOBANK_3** і встановіть **I/O standard** рівний **1.8 V**. Для шин адреси **read** і **write** призначте банк **IOBANK_3** і встановіть **I/O standard** рівний **1.8 V**.

1.11. У вікні **All Pins List** призначте сигналу **clock** контакт **Pin H16** і встановіть **I/O standard** рівний **1.8 V**.

1.12. У вікні **All Pins list** призначте сигналу **wren** банк **IOBANK_3** і встановіть **I/O standard** рівний **1.8V**.

2. Перевірка призначення контактам введення/виведення.

Коли зроблено основні призначення контактам введення/виведення, можна перевірити коректність цих призначень, без виконання повної компіляції проекту. Таким способом можна швидко і легко знайти помилки при призначеннях контактам введення/виведення і виправити їх. Для більш суворої перевірки призначень контактам введення/виведення використовується утиліта **I/O Assignment Analysis** спільно з утилітою **Live I/O Check.**

2.1. У меню **Processing** головного вікна **Quartus II**, в розділі **Start** виберіть команду **Start Analysis & Synthesis** або натисніть кнопку , розташовану в панелі інструментів головного вікна **Quartus II**. Натисніть **OK** після завершення компіляції.

Компілятор може відключити утиліту **Live I/O Check** і включити її пізніше. Якщо, після виконання компіляції, утиліта **Live I/O Check** не включенай, то її можна повторно включити у вікні **Pin Planner**.

2.2. У меню **Processing**, в розділі **Start** виберіть команду **Start I/O Assignment Analysis** чи натисніть кнопку в панелі інструментів **Pin Planner**. Натисніть **OK** після завершення аналізу.

Якщо результат аналізу не успішний, перевірте повідомлення у вікні **Messages** або повідомлення **Fitter Messages** у звіті компілятора **Compilation Report** (рис. 61).

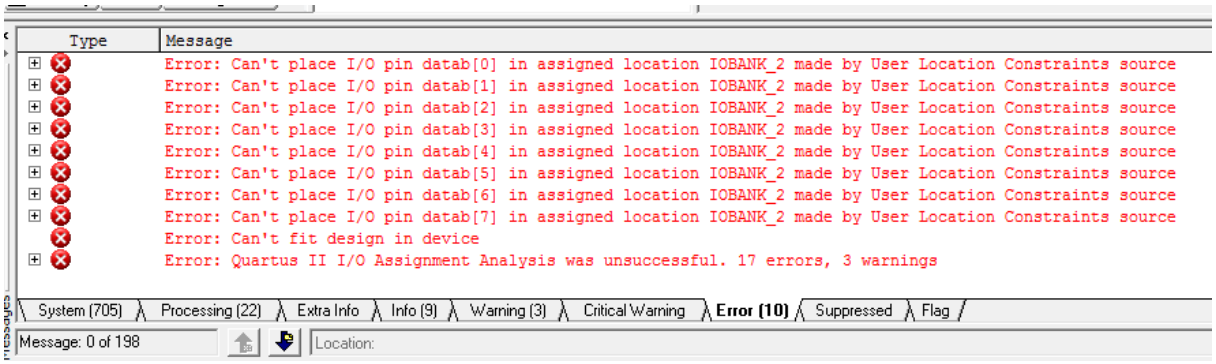


Рисунок 61. Вікно повідомлень про помилки призначення контактів введення/виведення

Утиліта **I/O Assignment Analysis** виявила помилки, про які видавалося повідомлення утилітою **Live I/O check** в процесі виконання призначень. Проаналізуйте повідомлення **I/O Analysis Messages** і визначте причину помилок. Розкрийте повідомлення про помилки для більш докладного аналізу причини того, чому всі сигнали шини **dataa** не можуть бути успішно розміщені.

Для визначення причин неможливості розміщення контактів введення/виведення необхідно: ретельний аналіз повідомлень про помилки і наявність знань про архітектуру сімейства **Cyclone II** (осередків введення/виведення ПЛІС).

2.3. Поверніться в **Pin Planner** і розкрийте групи сигналів **dataa** і **datab** у вікні **Groups list**. Якщо необхідно, змістіть сторінку, щоб бачити колонку **I/O Standard** у вікні **Groups list** або у вікні **All Pins list**.

Зверніть увагу, ви призначили для вхідних шин **dataa** і **datab** банк **I/O Bank 2**, але при цьому задали різні рівні напруги **VCCIO (1.8 & 2.5)**. ПЛІС Cyclone II, як і всі мікросхеми FPGA фірми Altera, працюють з єдиним рівнем напруги **VCCIO** для одного банку контактів введення/виведення **I/O bank**.

2.4. У вікні **Groups List** змініть призначення **I/O standard** для групи сигналів **datab** на **2.5 V**. Переконайтеся, що всі сигнали шини **datab** мають відповідне призначення. Зверніть увагу на зміни у вікні **Live I/O Check Status**.

2.5. Повторіть аналіз призначень за допомогою утиліти **I/O Assignment Analysis**. Натисніть **OK** після завершення.

Зверніть увагу, як швидко і легко можна перевірити призначення контактам введення/виведення без виконання повної компіляції проекту!

2.6. Щоб подивитися, як трасувальник розмістив призначені контакти введення/виведення, натисніть кнопку  в панелі інструментів **Pin Planner** або в меню **View** виберіть команду **Show -> Show Fitter Placements**. Натисніть **OK**, щоб відключити утиліту **Live I/O check**.

Утиліта **Live I/O check** повинна бути відключена при перегляді результатів розміщення. У вікні статусу з'явиться підтвердження про відключення.

2.7. Проведіть компіляцію проекту.

3. Збереження призначень контактів введення/виведення за допомогою утиліти Back-annotate.

Цей крок виконується після перевірки правильності призначень вихідних контактів, перед початком проектування друкованої плати. Тепер ви повинні переконатися, що розташування призначених контактів не змінилося при виконанні подальшої компіляції.

3.1. У меню **Assignments** виберіть команду **Back-Annotate Assignments**, щоб відкрити діалогове вікно **Back-Annotate Assignments** (рис. 62).

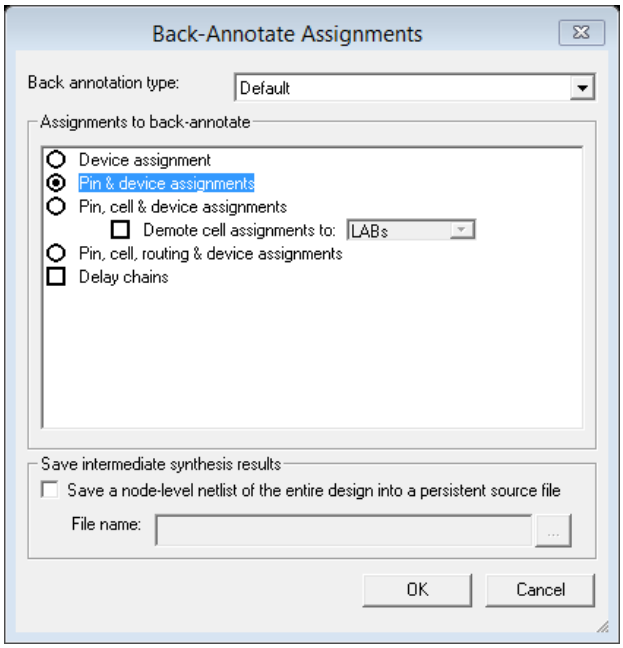


Рисунок 62. Діалогове вікно **Back-Annotate Assignments**

3.2. У розділі **Assignments to back-annotate**, відзначте опцію **Pin & device assignments** (зазвичай, встановлена за замовчуванням) як показано вище. Натисніть **OK**.

Зверніть увагу, всі зазначені зеленим кольором контакти введення/виведення, тепер ще мають червоне штрихування (рис. 63). Це

вказує на те, що контакти, які були призначені трасувальником, зараз ще є призначеними користувачем, і зберігаються як призначення у файлі **.QSF**.

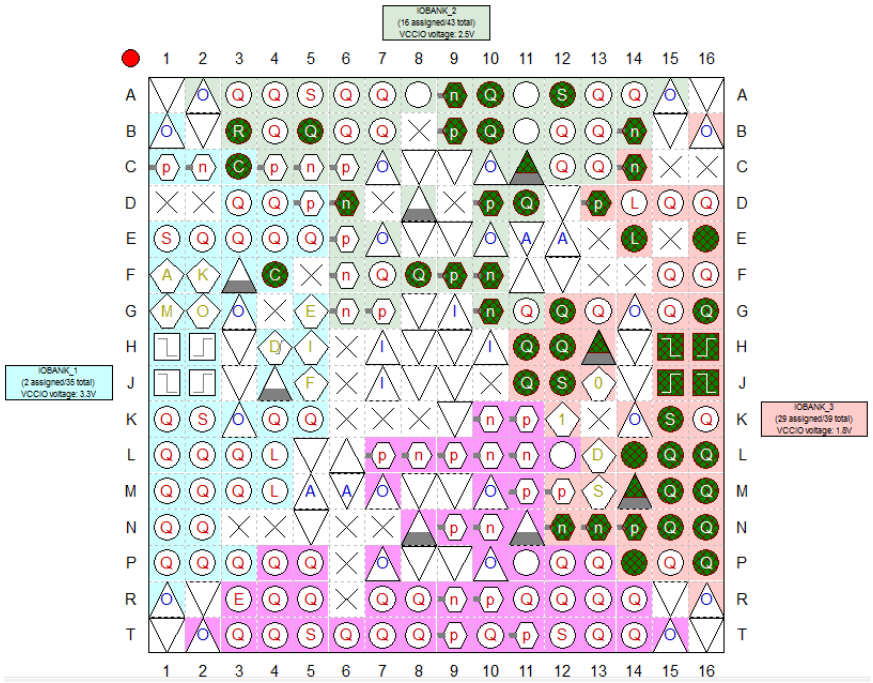


Рисунок 63. Зображення призначених контактів введення/виведення контактів

4. Перенесення призначення контактів введення/виведення у вихідну версію проекту.

Зараз ви повинні перенести призначення контактів вводу / виводу з поточної версії проекту у вихідну (з якою працювали раніше). Для цього, спочатку, ви збережете ваші призначення як файл формату **.CSV**, а потім імпортуєте їх у вихідну версію проекту.

4.1. Коли вікно **Pin Planner** активне, в меню **File** виберіть команду **Export**.

4.2. У діалоговому вікні **Export** задайте ім’я файлу **io_assignments.csv** і натисніть **Export**.

4.3. У меню **Assignments** виберіть команду **Import Assignments**. У діалоговому вікні **Import Assignments** натисніть кнопку пошуку в стрічки **File name:** оберіть файл **io_assignments.csv**. Відключіть опцію **Copy existing assignments into pipemult.qsf.bak before importing**. Натисніть **Open**, потім **OK**.

4.5. Відкрийте вікно **Pin Planner** і перевірте правильність перенесення призначень. Вимкніть режим перегляду контактів, розміщених трасувальником, за допомогою кнопки , або відповідної команди в меню **View** (розділ **Show**).

Завдання на лабораторну роботу 9

1. Ознайомитись із теоретичними відомостями.
2.
3.
4. Зробити висновки. Результати у вигляді скріншотів представити у звіті.

Контрольні питання

1.
2.
3.
4.
5.

Лабораторна робота № 10

Часовий аналіз проекту в середовищі Quartus II

Мета роботи: використовуючи утиліту TimeQuest навчитись виконувати часовий аналіз проекту в середовищі Quartus II.

Теоретичні відомості

Часовий аналіз у середовищі **Quartus II** може бути виконаний або класичним методом (**Classic Timing Analysis**) або за допомогою програми **TimeQuest**.

Часовий аналіз є складовою частиною повної компіляції. Результати часового аналізу дозволяють оцінити швидкодію проекту та виявити найбільш критичні ділянки, які потім можна оптимізувати схемотехнічно або задати їм інші опції компілятора. Повторна компіляція проекту покаже зміну часових параметрів змінених блоків.

Параметри часового аналізатора встановлюються за допомогою закладки **Classic Timing Analyzer Settings** діалогу **Settings** (меню **Assignments → Settings...**), яка показана на рисунку 64.

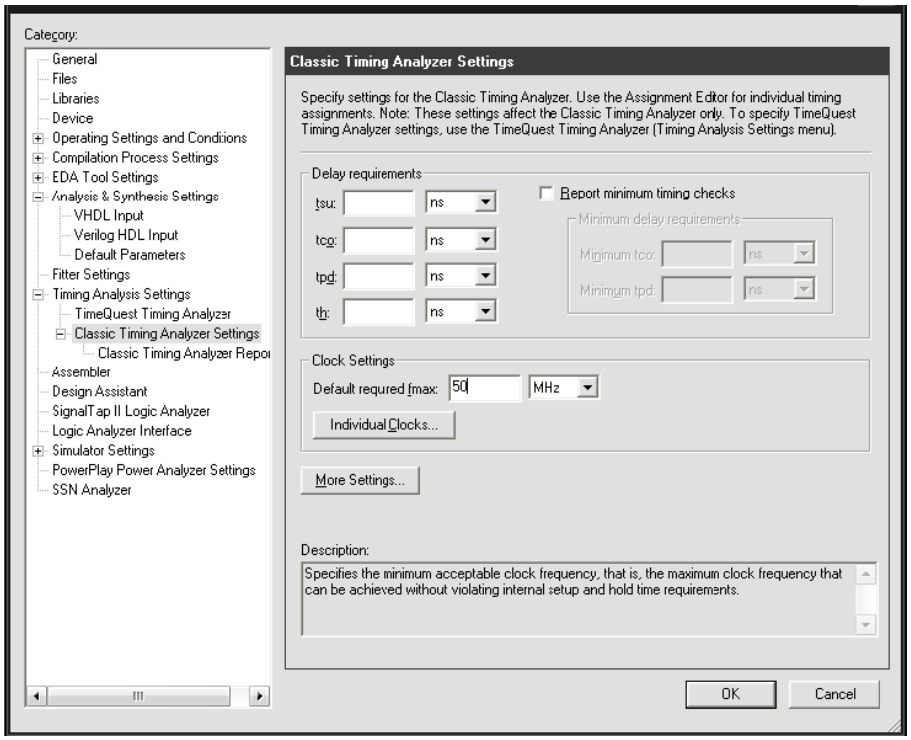


Рисунок 64. Встановлення часових параметрів

При роботі часовий аналізатор використовує наступні налагодження пакета:

- **fMAX** – максимальна тактова частота;
- **tSU** – час передвстановлення (**setup time**) для регістрів;
- **tPD** – час затримки проходження сигналу від входу через внутрішню комбінаційну логіку до виходу (**pin-to-pin delay**);
- **tCO** – затримка вихідного сигналу щодо тактової частоти (**clock to output delay**);
- **tH** – час утримання (**clock hold time**).

Обмеження часових параметрів числовими значеннями збільшує час компіляції, оскільки **Quartus II** спочатку виконує компіляцію проекту, потім перевіряє його на відповідність встановленим параметрам і, при наявності невідповідності, виконує оптимізацію проекту.

Для перегляду результатів часового аналізу необхідно відкрити вікно результатів компіляції та вибрати пункт **Timing Analyzer** (рис. 65).

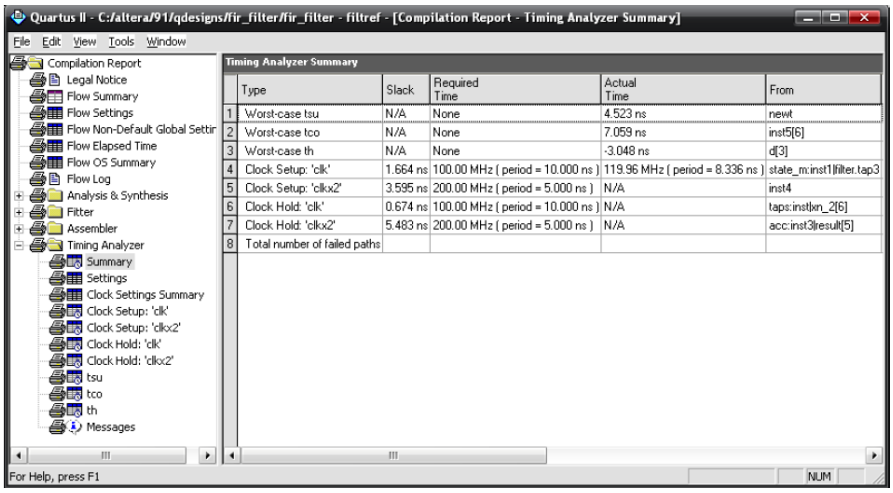


Рисунок 65. Результати часового аналізу

Дана закладка результатів компіляції містить наступні пункти:

Summary – загальне зведення результатів часового аналізу. Тут вказуються найгірші з можливих значень показників (**worst–case**) часу передустановки, утримання, максимальне значення тактових частот і т.п.

Settings – налагодження, які використовуються в процесі часового аналізу. Для їхньої зміни необхідно використовувати діалог **Settings.**

- **Clock Settings Summary** – збірка установок по всіх тактових частотах, які використовуються в проекті.
- **Clock Setup** – час предвстановлення сигналів даних, відносно сигналу тактової частоти. Відображається в полі **Slack.**

Максимально можлива частота для даного ланцюга відображається в полі **Actual fmax**.

- **Clock Hold** – час утримання сигналів даних відносно періоду тактової частоти.
- **tsu, tco, th** – часи передустановки, тактовий сигнал-вихід та утримання для обраних вузлів схеми.
- **Messages** – повідомлення, що з'являються в результаті часового аналізу.

Для додавання вузлів у таблицю часового аналізу необхідно вибрати пункт **Advanced List Paths...** у контекстному меню (рис. 66).

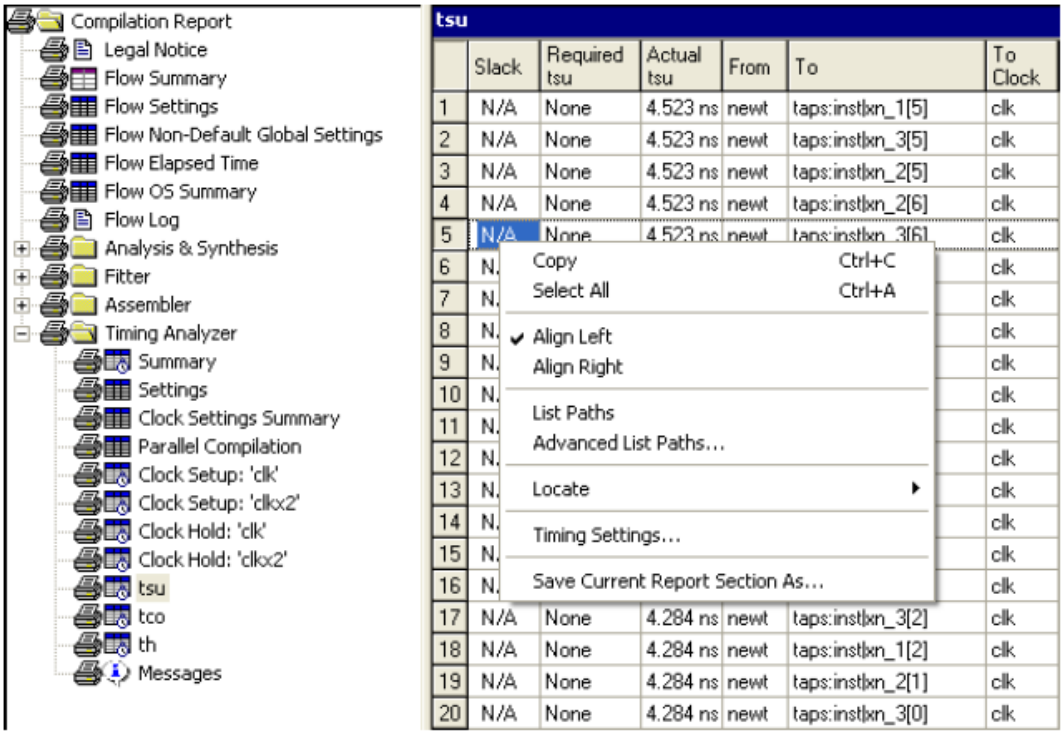


Рисунок 66. Перегляд результатів часового аналізу

Вікно, котре відкрилося (рис. 67) містить діалог, у якому необхідно вказати:

- Кількість шляхів, які додаються в схему – **Path to report**.
- Джерело сигналу (**From**) та його одержувач (**To**).
- Таблиці, у яких буде проводитися аналіз (**Report window sections**).

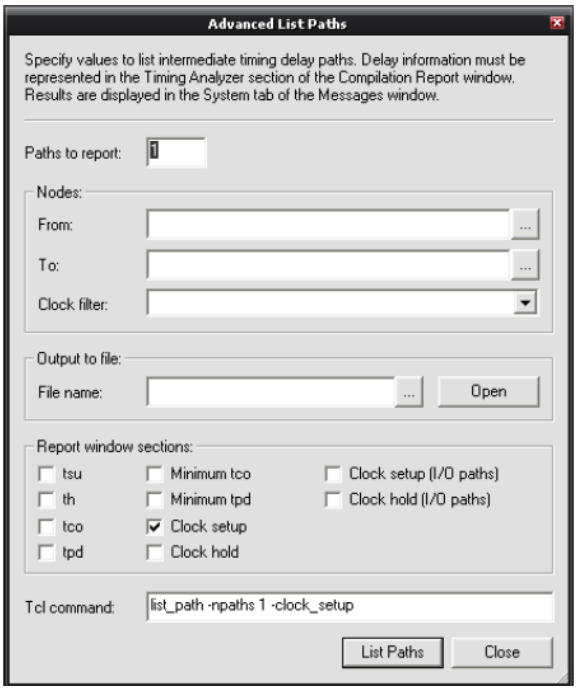


Рисунок 67. Додавання шляхів до часового аналізу

Хід проектування

1. Виконання синтезу проекту

1.1. Переконайтеся, що ви працюєте з основною версією проекту і натисніть кнопку для синтезу проекту.

Якщо раніше виконувалась повна компіляція проекту, повторний синтез дозволяє швидше створити список зв'язків проекту для установки призначень. І для компільованої версії проекту рекомендується виконати наступну послідовність дій.

2. Утиліта TimeQuest і створення списку зв'язків проекту для часового аналізу.

2.1. В основній панелі інструментів САПР **Quartus II** натисніть кнопку або, в меню **Tools** виберіть команду **TimeQuest Timing Analyzer**. Натисніть **No** для відмови від генерації файлу **.SDC** з файлу **.QSF**. Якщо ж випадково було натиснуто **Yes**, закрийте **TimeQuest**, відкрийте вікно **Explorer** і видаліть файл **pipemult.sdc** з робочої директорії проекту. Поверніться у **Quartus II** і повторно викличте утиліту **TimeQuest** (рис. 68).

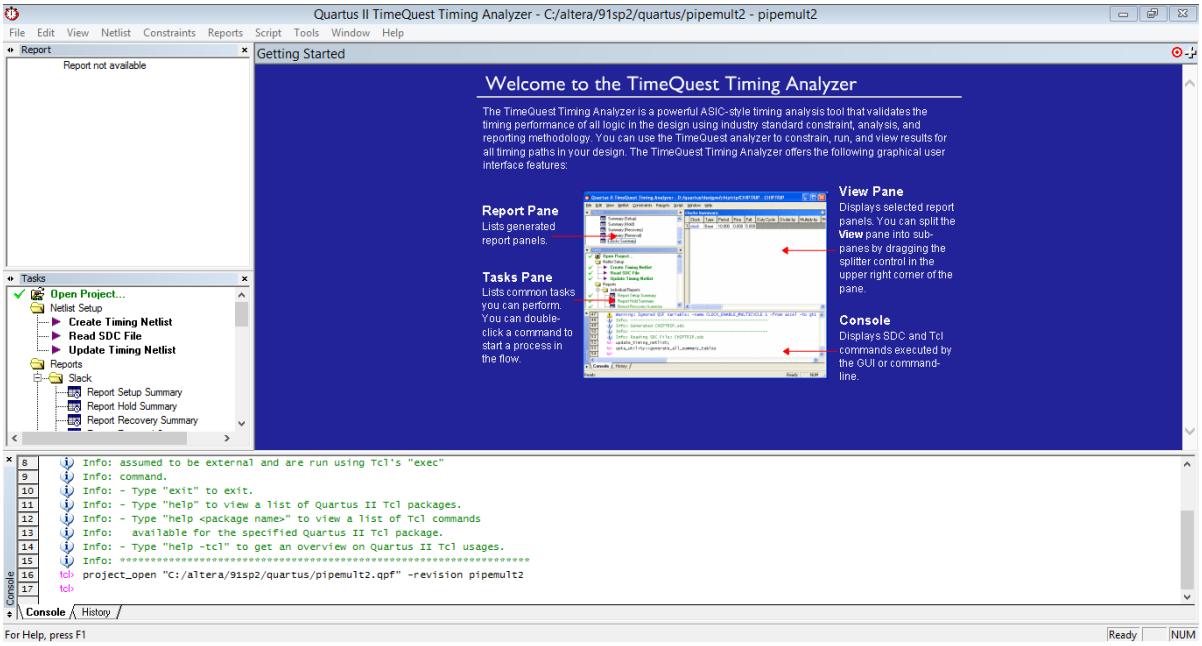
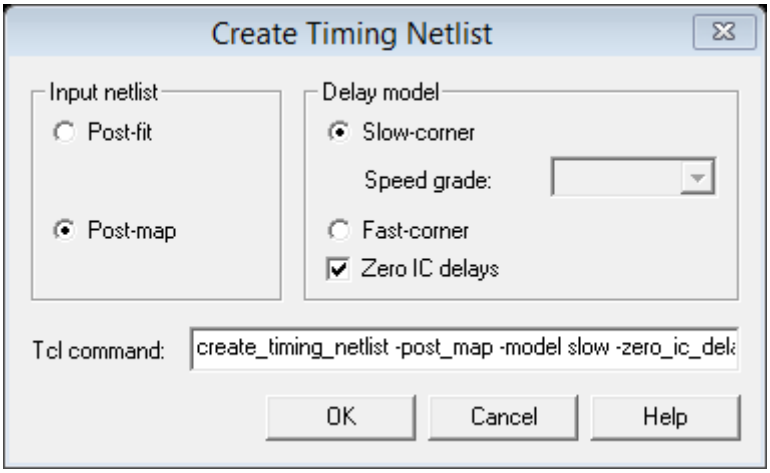


Рисунок 68. Вікно утиліти TimeQuest

2.2. Створення зв’язків проекту. для часового аналізу. У меню **Netlist** виберіть команду **Create Timing Netlist** і змініть тип вихідного списку **Input netlist** на **Post-map**, або в діалоговому вікні **Console** введіть команду **create_timing_netlist -post_map**. Натисніть **OK**.



Зелена мітка, що з'явилася поруч з командою **Create Timing Netlist** у вікні **Tasks**, вказує на успішне її виконання. Зверніть увагу, у вікні **Console** з'явилося повідомлення (виділене синім кольором) про те, що утиліта **TimeQuest** не є інструментом для виконання тимчасового аналізу проекту за замовчуванням. Це виправиться пізніше. Можна створити список зв'язків проекту подвійним натисканням мишки на команді **Create Timing Netlist** у вікні **Tasks**, але це не дозволило б вам змінити вихідний список зв'язків на **post-map**, так як він не задається за замовчуванням.

2.3. Відкрийте **.SDC** файл (команда **Read SDC File**). Або просто введіть команду **read_sdc** в рядку після запрошення **tcl>** діалогового вікна.

З'явиться повідомлення про те, що файл **.SDC** не був знайдений. Це правильно тому, що не вказувалось ім'я файлу, утиліта **TimeQuest** автоматично шукала будь які **.SDC** файли, додані до проекту, а з них - файли з таким же ім'ям, як і поточна версія проекту **pipemult**. Але таких файлів немає.

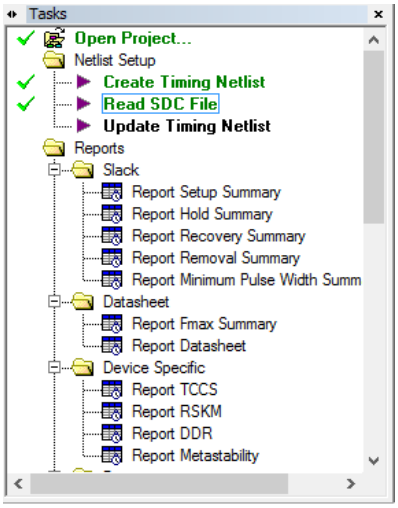


Рисунок 69. Вигляд вікна Task

Поруч з командою **Read SDC File** повинна з'явитися зелена мітка, яка вказує на спробу прочитати **.SDC** файл. Створіть цей файл за допомогою редактора **SDC** файлів (подібний текстовому редактору САПР **QuartusII**) утиліти **TimeQuest GUI**.

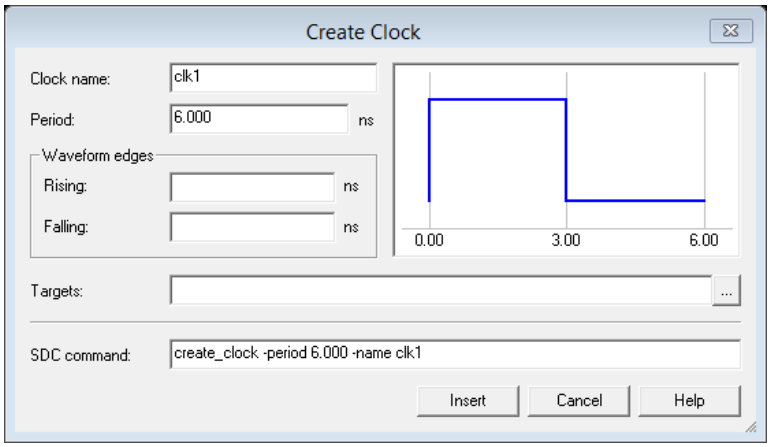
3. Створення файлу sdc і встановлення призначень для вхідної тактової частоти clock.

3.1. В меню **File** утиліти **TimeQuest** виберіть команду **New SDC File**. Відкриється вікно редактора **SDC** файлу для створення нового файлу.

3.2. У меню **File** редактора виберіть команду **Save As** і задайте ім'я файлу **pipemult.sdc**.

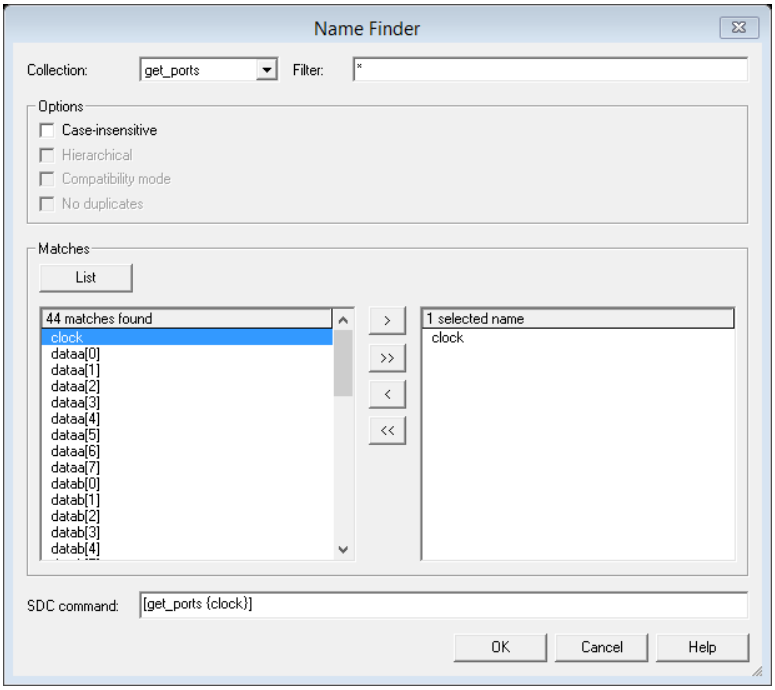
3.3. За допомогою команди **Create Clock** задайте період тактової частоти, рівний **6 ns** для вхідного сигналу **clock**.

У меню **Edit** редактора **SDC** файлів в розділі **Insert Constraint** оберіть команду **Create Clock ...** .У діалоговому вікні **Create Clock** задайте ім'я сигналу тактової частоти **clock**. У рядку **period** встановіть значення **6**.



Зверніть увагу, можна змінити розташування фронтів тактового сигналу, тобто створити сигнал зі шпаруватістю, яка не рівна 50%. Залишаємо значення за замовчуванням.

3.4. У рядку targets натисніть кнопку [...]. У вікні **Name Finder**, виберіть **get_ports** з випадаючого меню **Collection**. У розділі **Matches** натисніть кнопку **List**. Двічі натисніть кнопку мишки на сигналі **clock** в сформованому списку. Натисніть **OK**



3.5. Натисніть кнопку **Insert**. Збережіть файл. Призначення тактової частоти добавлено в **.SDC** файл в місці розташування курсору. Далі оновіть список зв'язків проекту для часового аналізу для того, щоб переконатися, що створене призначення допустиме.

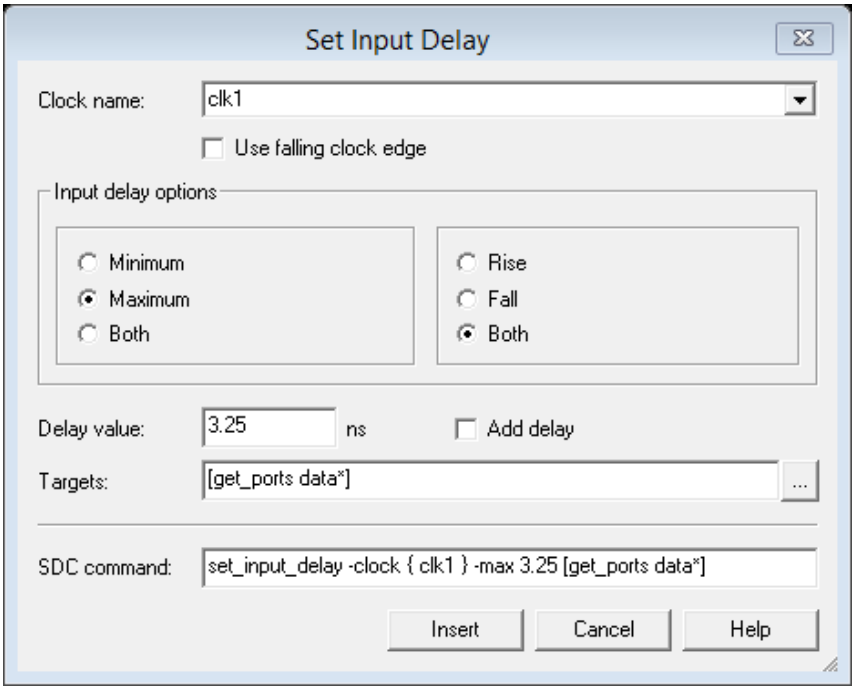
3.6. Поверніться в головне вікно **TimeQuest** і двічі натисніть лівою кнопкою мишки на команді **Update Timing Netlist** у вікні **Tasks**. Поверніться або повторно відкрийте ваш **SDC** файл.

4. Додавання тимчасових призначень для контактів введення/виведення.

4.1. У новому рядку файлу, за допомогою команди **Set Input Delay**, встановлюється максимальна вхідна затримка для двох 8-розрядних шин **dataa** і **datab** рівну **3.25 ns** щодо тактової частоти **clk1**.

У меню **Edit**, в розділі **Insert Constraint** виберіть команду **Set Input Delay**. Введіть наступні значення:

- **Clock name = clk1** (використовуйте випадаюче меню)
- **Input Delay Options = Maximum** (виберіть **Both** для опції **Rise \ Fall**)
- **Delay Value = 3.25**
- **Targets = [get_ports data *]**



Можете скористатися кнопкою перегляду і утилітою **Name Finder** для вибору всіх сигналів вхідних даних, але іноді, особливо при роботі з шинами, зручніше задавати тип в командному рядку і використовувати символи узагальнення. Якщо імена групи сигналів невідомі, можна скористатися утилітою **Name Finder** і додати символи узагальнення в командному рядку **Name Finder SDC**, вибравши сигнал, що належить відповідній групі.

4.2. За допомогою команди **Set Input Delay**, встановіть мінімальну вхідну затримку рівну **1.75 ns** для двох 8-розрядних шин **dataa** і **datab** щодо тактової частоти **clk1**. Щоб зробити це швидше можна, використовуючи команди **copy** і **paste**, скопіювати рядок з командою установки максимальної вхідної затримки і вставити її в новому рядку. Відредагувати новий рядок, змінивши **-max** на **-min** і **3.25** на **1.75**.

4.3. Подібним чином, за допомогою команди **Set Output Delay**, встановлюється максимальне значення вихідної затримки, рівне **0.7 ns** для 16-розрядної вихідний шини даних **q**, щодо тактової частоти **clk**1.

За допомогою команди **Set Output Delay**, встановіть мінімальне значення вихідний затримки рівне **0.0 ns** для 16-розрядної вихідний шини **q**, щодо тактової частоти **clk1**.

4.4. За допомогою команди **Set Input Delay**, встановіть всі параметри для вхідних сигналів **wren**, **rdaddress** і **wraddress**. Значення максимальної вхідний затримки має бути рівне **2.5 ns**, мінімальна вхідна затримка – **1.0 ns**, щодо тактової частоти **clk1**. Збережіть файл (рис. 70).

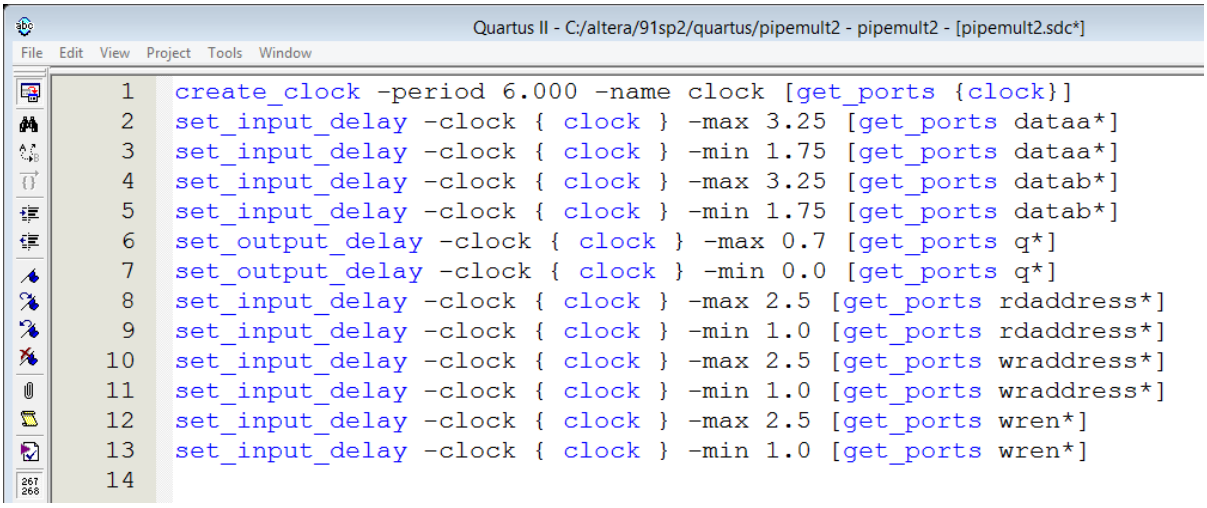


Рисунок 70. Вікно тимчасових призначень для контактів вводу/виводу

5. Оновлення списку зв'язків проекту для часового аналізу

Для того, щоб нові обмеження набули чинності, необхідно оновити список зв'язків проекту для часового аналізу. У меню **Netlist** основного вікна утиліти **TimeQuest** виберіть команду **Delete Timing Netlist**.

Повторно зайдіть в меню **Netlist** і створіть новий список зв'язків проекту на основі списку **Post-map**. Оновіть список зв'язків проекту з урахуванням нових призначень, які були додані в **SDC** файл. У вікні **Tasks**

(утиліти **TimeQuest**), двічі натисніть ліву кнопку мишки на команді **Update Timing Netlist**.

Можна виконати команду **Read SDC File** вручну, перед оновленням списку зв'язків проекту, але при виконанні команди оновлення списку зв'язків проекту автоматично зчитується створений файл **SDC** з ім'ям, що збігається з ім'ям проекту.

6. Перевірка правильності встановлених значень за допомогою звітів утиліти TimeQuest

Після оновлення списку зв'язків проекту, можна створити різні звіти. Перше, що необхідно зробити – це перевірити правильність всіх встановлених призначень.

6.1. У вікні **Tasks** двічі натисніть ліву кнопку мишки на команді **Report SDC**. У вікні **Report** буде створена нова папка **SDC Assignments**, що містить три звіти: **Create Clock**, **Set Input Delay** і **Set Output Delay**.

6.2. У вікні **Tasks** двічі натисніть ліву кнопку мишки на команді **Report Clocks**. Даний звіт використовується для перевірки правильності призначення тактової частоти (тактових частот) та їх прив'язки до відповідних портів або контактам.

6.3. У вікні **Tasks** двічі натисніть ліву кнопку мишки на команді **Report Ignored Constraints**.

Цей звіт містить ті призначення, які були проігноровані утилітою **TimeQuest**. Наприклад, якщо некоректно вказано ім'я порту, то така **SDC** команда буде проігнорована і інформація про це з'явиться у цьому звіті. Якщо в даному звіті є інформація про неправильно зроблені призначення, то потрібно відкрити **SDC** файл, виправити помилку, повторно створити і обновити список зв'язків проекту і згенерувати даний звіт.

6.4. У вікні **Tasks** двічі натисніть ліву кнопку мишки на команді **Report Unconstrained Paths**.

Даний звіт використовується для перевірки повноти зроблених призначень в проекті.

6.5. Звіт **Unconstrained Paths Summary** (зазначений червоним кольором) вказує, що один вхідний порт і 16 зв'язків вхідного порту залишилися без призначень. Відкрийте папки **Setup Analysis** і **Hold Analysis**, щоб перевірити який саме це порт і які зв'язки.

7. Створення командного файлу Tcl для автоматичного виконання команд.

Оновлення списку зв'язків проекту і звітів кожного разу після зміни **SDC** файлу, процес досить трудомісткий. Для його автоматизації створюється простий командний файл на мові **Tcl**, який буде оновлювати всі дані після зміни **SDC** файлу.

7.1. В основному вікні утиліти **TimeQuest** створіть новий **SDC** файл. У меню **File** виберіть команду **Save As**. Задайте ім'я файлу **pipemult_timing.tcl** і виберіть настройку **Save as type** як **Tcl Script File (*.tcl)**.

Оскільки редактор **SDC** файлів і текстовий редактор **Quartus II** однакові, можна створити файл на мові **Tcl**, не переходячи в основне вікно САПР Quartus II.

7.2. Введіть команди мовою **Tcl** в новий файл, у зазначеному нижче порядку. Ви можете вводити кожну команду вручну або копіювати їх з діалогового вікна утиліти **TimeQuest GUI** (закладка **Console** або закладка **History**).

7.3. Видаліть існуючий список зв’язків проекту. Створіть список зв'язків проекту на основі **post-map** списку. Зчитайте файл **SDC**. Оновіть список зв'язків проекту для часового аналізу. Створіть звіт про призначення для тактової частоти. Створіть звіт про сигнали без призначень. Створіть звіт про проігнорованих призначення. Створіть звіт про внесені **SDC** призначення. Збережіть файл (рис. 71).

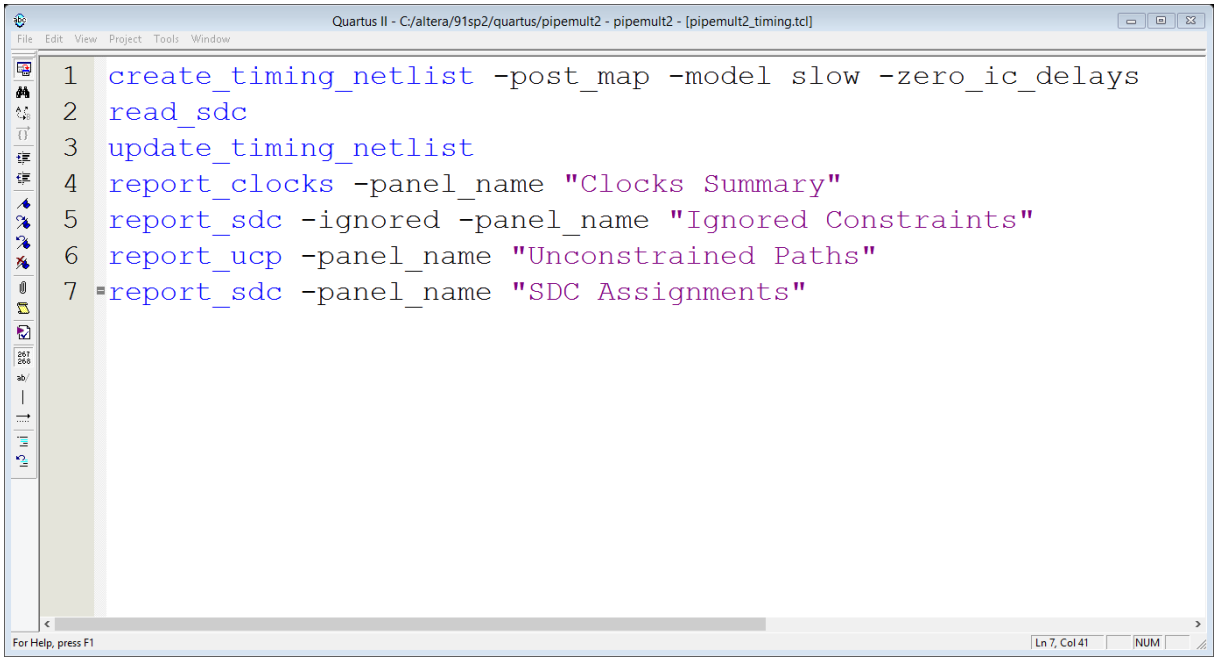


Рисунок 71. Створення файлу для автоматичного оновлення зв’язків проекту

7.4. В основному вікні **TimeQuest**, увійдіть в меню **Netlist** і видаліть поточний список зв'язків проекту. У меню **Script** виберіть команду **Run Tcl Script** і вкажіть командний файл **pipemult_timing.tcl**.

Автоматично повинен бути створений і оновлений список зв'язків проекту і повинні бути відновлені всі звіти (рис. 72). Єдиним недоліком, в даному випадку, є необхідність вручну видаляти список зв'язків проекту до запуску командного файлу. Інакше утиліта **TimeQuest** видаватиме повідомлення про помилку. Якщо ввести команду видалення списку зв'язків проекту на початку командного файлу, може виникнути інша проблема: якщо список зв'язків проекту ще не створений, утиліта **TimeQuest** видаватиме інше повідомлення про помилку. Можна створити більш складний командний файл, що враховує цю ситуацію.

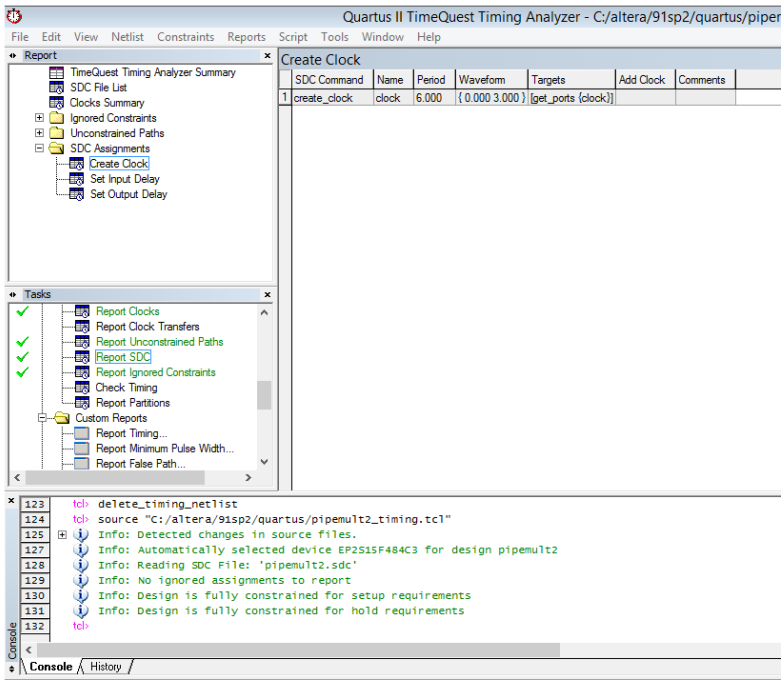


Рисунок 72. Запуск файлу pipemult_timing.tcl

8. Виконання компіляції проекту, з урахуванням даних SDC файлу.

8.1. Перейдіть до основного вікна САПР Quartus II. У меню **Assignments** виберіть команду **Timing Analysis Settings**. Відкриється діалогове вікно **Settings** з обраним розділом **Timing Analysis Settings** (рис. 73).

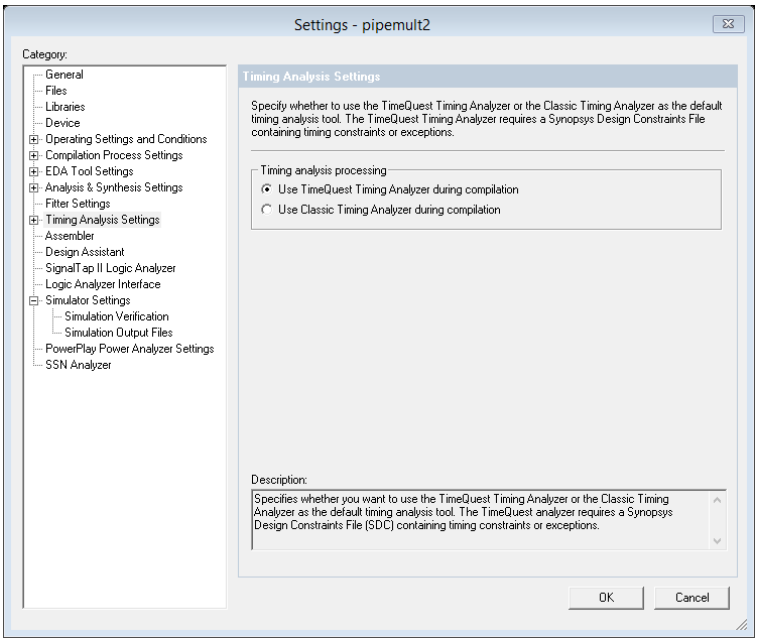
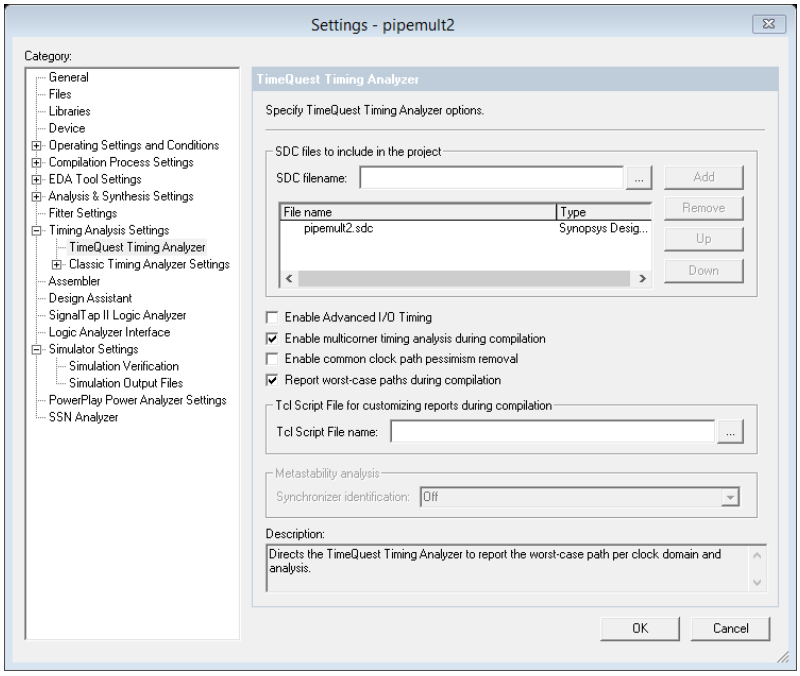


Рисунок 73. Діалогове вікно Settings

Відмітьте функцію **Use TimeQuest Timing Analyzer during compilation**.

8.2. Додайте файл **pipemult.sdc** до проекту. У діалоговому вікні **Settings** виберіть категорію **TimeQuest Timing Analyzer** (у розділі **Timing Analysis Settings**). За допомогою кнопки ... виберіть файл **pipemult.sdc**, натисніть кнопку **Open**, а потім кнопку **Add**.

8.3. Встановіть опції **Enable multicorner timing analysis during compilation** і **Report worst-case paths during compilation**.



8.4. Натисніть **OK**, щоб закрити діалогове вікно **Settings**.

8.5. Натисніть кнопку  або виберіть команду **Start Compilation** в меню **Processing**.

8.6. Після завершення компіляції, відкрийте папку **TimeQuest Timing Analyzer** у звіті компілятора **Compilation Report**.

8.7. Випишіть найгірші значення тимчасового запасу (**slack**) для часу установки (**setup**) і часу утримання (**hold**) сигналу. Пам'ятайте, що ці значення можуть бути різними для швидкої і повільної моделі.

9. Застосування SDC файлу до версії проекту pipemult_lc

Для перевірки тимчасових характеристик версії проекту **pipemult_lc** необхідно скористатися створеним **SDC** файлом.

9.1. За допомогою випадаючого меню у верхній частині основного вікна **Quartus II** змініть поточну версію проекту на **pipemult_lc**.

9.2. Застосуйте утиліту **TimeQuest** в процесі компіляції (**Assignments-> Timing Analysis Settings**).

9.3. Додайте файл **pipemult.sdc** до проекту в якості файлу опису тимчасових призначень, встановіть опції **Enable multicorner timing analysis during compilation** і **Report worst-case paths during compilation**.

9.4. Виконайте повну компіляцію версії проекту **pipemult_lc**.

9.5. У звіті компілятора **Compilation Report** відкрийте таблиці **Setup Summary** і **Hold Summary**, відповідні швидкої і повільної тимчасовим моделями. Випишіть найгірші значення тимчасового запасу (**slack**) для часу установки (**setup**) і часу утримання (**hold**) сигналу в Таблицю.

Видно, що для сигналу **clk1** не виконується вимога до часу установки, рівне **0.7 ns**.

9.6. Відкрийте утиліту TimeQuest.

У даному випадку, оскільки SDC файл вже створений і доданий до поточного проекту, в ньому повністю описані всі тимчасові призначення, то можна перейти у вікно **Tasks** для генерації звітів. Додатково, можна відредагувати файл **pipemult_timing.tcl** для створення списку зв'язків проекту на основі **post-fit** списку для швидкої чи повільної часової моделі і генерації звітів про тимчасові запасах для часу установки й утримання.

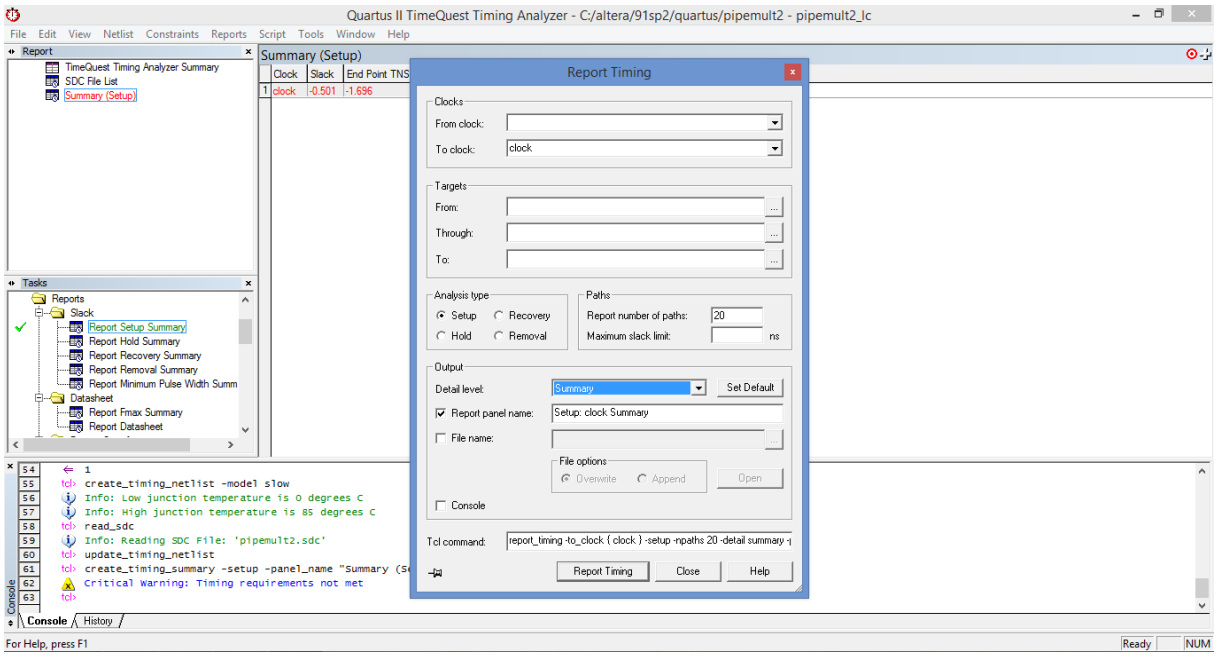
9.7. Згенеруйте звіт **Setup Summary**. У вікні **Tasks** двічі натисніть ліву кнопку мишки на команді **Report Setup Summary**.

Зверніть увагу, поряд з командами **Create Timing Netlist, Read SDC File** і **Update Timing Netlist** з'явилися відповідні зелені позначки про їх

виконання. Виклик команди генерації звітів з вікна **Tasks** автоматично виконує всі необхідні команди для їх створення. Така можливість існує для даного проекту тому, що він використовує заданий за замовчуванням **post-fit** список для створення списку зв'язків проекту, на відміну від **post-map** списку, який було обирана вручну на попередніх кроках.

У вікні **Report** звіт **Summary** (**Setup**) відзначений червоним кольором, повідомляючи про наявність помилки (яка вже відома).

9.8. Створіть докладний звіт. У звіті **Summary (Setup)** виділіть сигнал **clk1**. Натисніть праву клавішу мишки і виберіть з контекстного меню команду **Report Timing**.



9.9 У діалоговому вікні **Report Timing**, в розділі **Paths** встановіть значення **Report number of paths** дорівнює 20. У розділі **Output** для вказівки **Detail level:** за допомогою випадного меню виберіть значення **Summary**. Натисніть кнпку **Report Timing**.

З'явився звіт **Setup: clk1 Summary**. Всі логічні елементи, що реалізують помножувач, розташовані між двома банками регістрів, вхідним і вихідним. У результаті реалізації помножувача на логічних осередках, між фіксуючими регістрами розташовується велика кількість логічних елементів, що погіршує швидкодію пристрою. Щоб переконатися в цьому, скористайтеся утилітою **Technology Map Viewer** для перегляду використовуваних ресурсів.

9.10. Поверніться в основне вікно **Quartus II** і відкрийте утиліту **Technology Map Viewer** з меню **Tools.**

9.11. Натисніть праву клавішу мишки на будь-якому блоці і виберіть команду **Viewer Options** з контекстного меню. У розділі **Filtering** відключіть опцію **Number of filtering levels**. Натисніть **OK.**

9.12. Перейдіть нижче на два ієрархічних рівня в блоці **mult.**

9.13. Виділіть вихідний контакт **OUT1**. Натисніть праву клавішу мишки і виберіть команду **Filter / Sources**. Ви повинні побачити регістр з ім'ям **output_reg [0]**. Це вихідний регістр помножувача. Між вихідним регістром помножувача і вихідним контактом немає логічних елементів.

9.14. Знову виділіть **output_reg [0]**, натисніть праву клавішу мишки і виберіть команду **Filter / Sources.**

Тепер видно два рівні логічних елементів (реалізують функцію множення) і три вхідних регістри.

9.15. Виділіть один з трьох регістрів, натисніть праву клавішу мишки і виберіть команду **Filter / Sources.**

Зараз видно вхідний регістр даних (**SDATA**), який безпосередньо з'єднується з вхідним контактом. Таким чином, трасувальник розмістив усі логічні елементи помножувача між двома банками вхідних і вихідних регістрів.

10. Виконання фізичного синтезу для нової версії проекту

Перш, ніж скористатися можливістю фізичного синтезу для поліпшення часових параметрів, створіть нову версію проекту. Це корисно тому, що якщо проект вимагає тривалого часу для компіляції а результат фізичного синтезу виявиться незадовільним, можна швидко повернутися до попередньої версії проекту.

10.1. У меню **Project** основного вікна САПР QuartusII виберіть команду **Revisions.**

10.2. У діалоговому вікні **Revisions** натисніть кнопку **Create.**

10.3. Вкажіть ім'я **name_lc_phys_syn** для нової версії **Revision name**. Інші налаштування залиште без змін. Натисніть **OK.**

10.4. Натисніть **OK**, щоб закрити діалогове вікно **Revisions.**

10.5. У меню **Assignments** виберіть команду **Settings.** Перейдіть на сторінку **Physical Synthesis Optimizations** в розділі **Fitter Settings**.

10.6. У вікні **Fitter optimizations** встановите опцію **Perform register retiming**. У разділі **Physical synthesis effort** встановіть режим **Normal** (рис. 74).

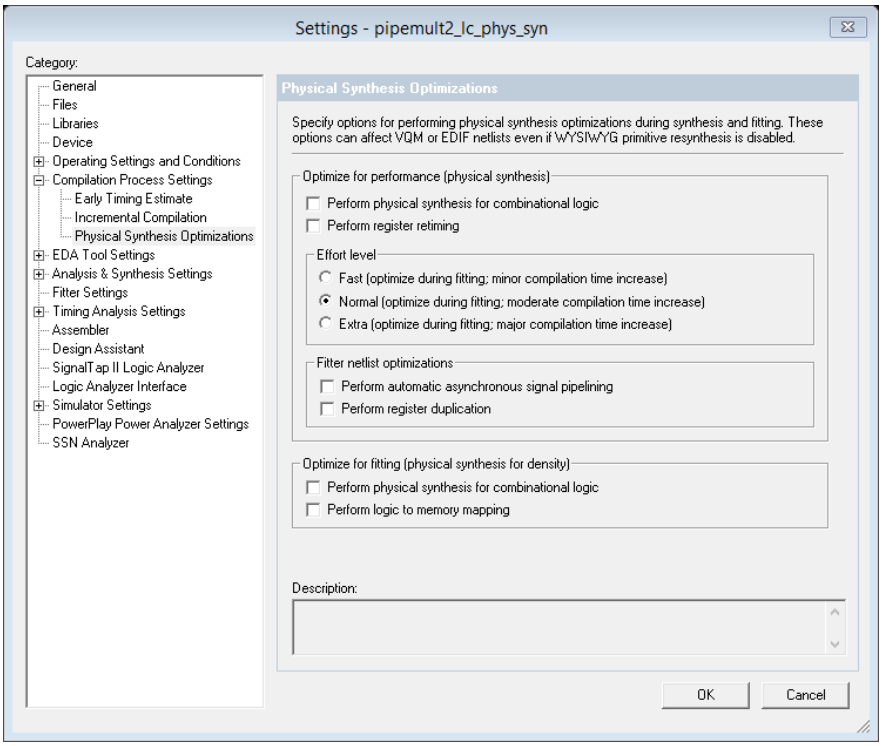


Рисунок 74. Вікно **Fitter optimizations**

10.7. Натисніть **OK**, щоб закрити діалогове вікно **Settings**.

10.8. Натисніть кнопку  для виконання повної компіляції.

Створюючи нову версію проекту, засновану на попередній версії всі необхідні налаштування (файл SDC, призначення контактів введення / виводу і ін.) переносяться з базової версії, тому немає необхідності їх задавати.

10.9. Після завершення компіляції, перевірте виконання тимчасових вимог. Запишіть отримані результати в Таблицю 3.

Зараз отримано дві різні версії проекту, що задовольняють вимогам до тимчасових характеристик. Нагадуємо, якщо у вашому проекті присутні помножувачі, то варіант їх реалізації за замовчуванням (з використанням спеціалізованих ресурсів) дає найбільш швидкодіючий результат з мінімальними витратами логічних осередків. Якщо потрібна реалізація помножувача на логічних елементах, необхідно зробити деякі кроки для оптимізації вашого проекту.

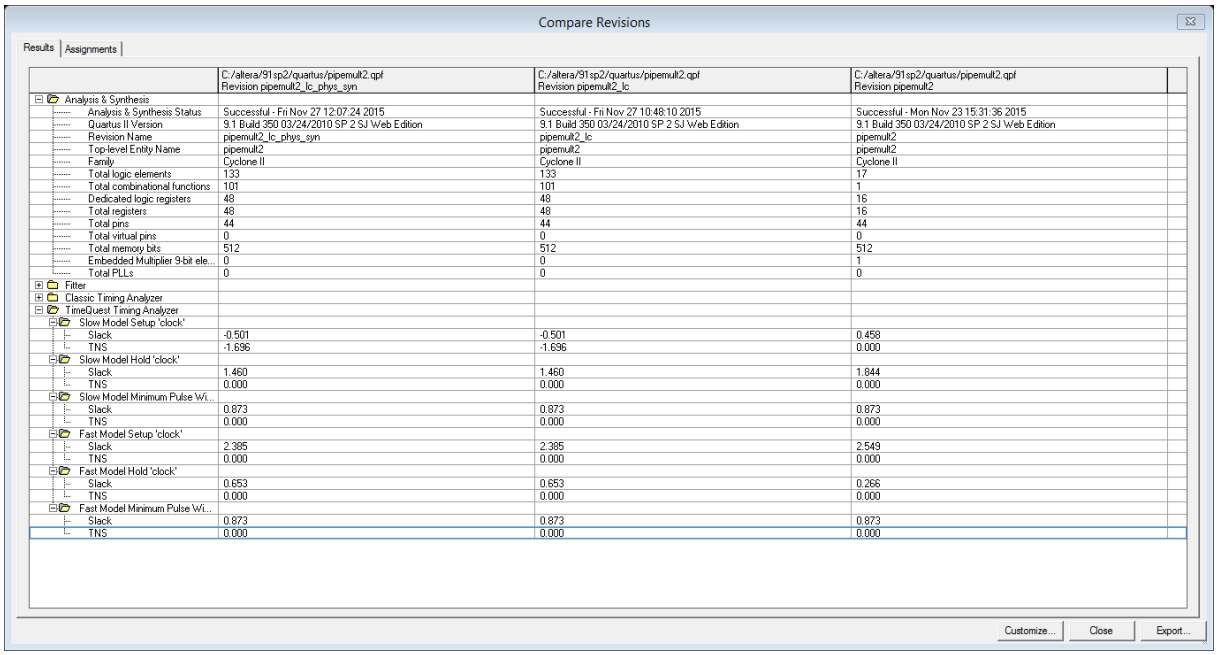
11. Порівняння отриманих результатів для різних версій проекту

11.1. У меню **Project** виберіть команду **Revisions**.

11.2. У діалоговому вікні **Revisions** натисніть кнопку **Compare**.

Відкриється таблиця порівняння, в якій відображені зроблені призначення та отримані результати для трьох версій проекту, створених вами (рис.75). Порівняйте:

- Загальна кількість логічних елементів – **Total number of logic elements (Fitter section)**
- Загальна кількість регістрів – **Total number of registers (Fitter or Analysis & Synthesis section)**
- Загальна кількість спеціалізованих блоків множення – **Total embedded multiplier 9-bit elements (Fitter section)**
- Часовий запас – **Setup & hold slack (TimeQuest section)**



	C:/altera/91sp2/quartus/pipemult2.qpf Revision pipemult2_lc_phys_syn	C:/altera/91sp2/quartus/pipemult2.qpf Revision pipemult2_lc	C:/altera/91sp2/quartus/pipemult2.qpf Revision pipemult2
Analysis & Synthesis			
Analysis & Synthesis Status	Successful - Fri Nov 27 12:07:24 2015	Successful - Fri Nov 27 10:48:10 2015	Successful - Mon Nov 23 15:31:36 2015
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	pipemult2_lc_phys_syn	pipemult2_lc	pipemult2
Top-level Entity Name	pipemult2	pipemult2	pipemult2
Family	Cyclone II	Cyclone II	Cyclone II
Total logic elements	133	133	17
Total combinational functions	101	101	1
Dedicated logic registers	48	48	16
Total registers	48	48	16
Total pins	44	44	44
Total virtual pins	0	0	0
Total memory bits	512	512	512
Embedded Multiplier 9-bit ele...	0	0	1
Total PLLs	0	0	0
Fitter			
Classic Timing Analyzer			
TimeQuest Timing Analyzer			
Slow Model Setup 'clock'			
Slack	-0.501	-0.501	0.458
TNS	-1.696	-1.696	0.000
Slow Model Hold 'clock'			
Slack	1.460	1.460	1.844
TNS	0.000	0.000	0.000
Slow Model Minimum Pulse Wi...			
Slack	0.873	0.873	0.873
TNS	0.000	0.000	0.000
Fast Model Setup 'clock'			
Slack	2.385	2.385	2.549
TNS	0.000	0.000	0.000
Fast Model Hold 'clock'			
Slack	0.653	0.653	0.266
TNS	0.000	0.000	0.000
Fast Model Minimum Pulse Wi...			
Slack	0.873	0.873	0.873
TNS	0.000	0.000	0.000

Завдання на лабораторну роботу 10

1. Ознайомитись із теоретичними відомостями.
2.
3.
4. Зробити висновки. Результати у вигляді скріншотів представити у звіті.

Контрольні питання

1.
2.
3.
4.

5.

Список використаних літературних джерел

1. Сергиенко А.М. VHDL для проектирования вычислительных устройств / А.М. Сергиенко – ТИД "ДС", 2003. – 208с.

2. Тарасов И.Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL / И.Е. Тарасов – Горячая Линия – Телеком, 2005. – 256 с.

3. Brown S. Fundamentals of Digital Logic with VHDL (2nd edition) / S. Brown Z. Vranesic – McGraw-Hill, 2004. – 939 p.